Concurrency in Interaction Nets and Graph Rewriting

Andrei Dorman

Tesi di dottorato in
Filosofia e Teoria delle Scienze Umane
Università degli Studi Roma Tre

Thèse de doctorat en
Informatique
Université Paris 13 - Sorbonne Paris Cité

Concurrency in Interaction Nets and Graph Rewriting

Andrei Dorman

Under the supervision of:

Damiano Mazza
Université Paris 13

Lorenzo Tortora de Falco
Università Roma Tre

Stefano Guerini
Université Paris 13

Referees:

Maribel Fernández
Catuscia Palamidessi

Jury:

Fabio Gadducci
Stefano Guerrini
Daniel Hirschkoff
Cosimo Laneve
Damiano Mazza
Lorenzo Tortora de Falco

Merci

I would like to thank my co-supervisor Damiano Mazza for his permanent guidance and support. He introduced me to pretty much any subject present in this thesis and even more, since I was unaware of Linear Logic before I started to work with him for my master's thesis, had no idea of the existence of neither Curry nor Howard and did not know a semantics could be denotational. Back then, I was a simple ol' admirer of Gödel and had no idea logic could go in such practical yet beautiful directions.

I also wish to thank him and Tobias Heindel for agreeing to co-author articles with me. I have learned a lot at their contact, as much in the process of writing as in the one of research, alone but most of all in collaboration. If some parts of this thesis are nicely written, it is most likely their accomplishment. Thanks to Tobias, I have discovered parts of Germany I would probably have not otherwise.

A broad thanks to the team LCR of the LIPN for their warm welcome and the extraordinary way they take care of their PhD students. In general, the working environment of the "Franco-Italian linear logic community" is very stimulating, thanks to their professionalism but also their kindness and support. Every opportunity is given to learn, exchange and discuss.

Thank you in advance to the referees Maribel Fernández and Catuscia Palamidessi whose work has inspired some of the tools and results of this thesis. I hope they do not realize that I somehow misunderstood their work.

A word to all the PhD students that I have met again and again throughout these three years in various locations of the world, with which we have exchanged about work, but not only, and who have made every conference, summer and winter school particularly enjoyable. In order not to expose their extra-professional abilities, I shall not thank them namely, but they will recognize themselves.

Thanks you of course to Lorenzo who discovered me, wandering, confused and lost, in the hallways of Chevaleret. He made possible for me to do this graduate program and has given me the opportunity to fulfill my long-time dream of living in Rome.

Contents

Introduction **1**
Plan of the thesis 16

1 Interaction Nets **19**
1.1 Lafont's calculus 19
1.1.1 Interaction nets 19
1.1.2 Universality 25
1.2 Non-deterministic extensions 29
1.2.1 Multirule nets 29
1.2.2 Multiwire nets 30
1.2.3 Multiport nets 31
1.3 Relative expressivity 33
1.3.1 Encodings among non-deterministic interaction nets 33
1.3.2 A separation technique based on Event Structures 37
1.3.3 Event Structures of multiport vs. multiwire systems 40

2 Structural Operation Semantics for Interaction Nets **43**
2.1 SOS for simple interaction nets 43
2.2 Graph rewriting : preliminaries 46
2.2.1 Hypergraphs 46
2.2.2 Standard graph transformation 47
2.2.3 Behavior as interaction with the environment 50
2.3 A process calculus perspective on borrowed contexts 52
2.3.1 The analogy with CCS 53
2.3.2 Borrowed contexts in three layers 55
2.4 Communication in composed states 61
2.4.1 The idea of composition of transitions 61
2.4.2 Composition results for Borrowed Context diagrams 63
2.5 SOS semantics 70
2.6 Application to interaction nets 73
2.6.1 Towards a partial solution 76
2.6.2 Sufficient conditions 78

2.6.3 Particular means for particular ends . . . 83
2.7 Conclusion . . . 87

3 Concurrent Interaction Nets **91**
3.1 (Textual) interaction nets . . . 91
3.2 Encoding the pi-calculus . . . 97
3.3 Comparing interaction net extensions . . . 102
3.3.1 Encodability . . . 103
3.3.2 Separation . . . 105
3.3.3 To sum-up . . . 106
3.4 Multiports can alone express rule ambiguity and connectors . . . 107
3.4.1 Uniports, but multwires and/or multiple rules . . . 107
3.4.2 Some issues with communication zones . . . 115
3.4.3 Upgrade to multiport source language . . . 118
3.4.4 Encoding general nets into multiport nets . . . 121
3.5 Multiwires can express rule ambiguity . . . 128
3.5.1 One multirule to rule them all . . . 129
3.5.2 Encoding the asymmetric rule using multiwires . . . 133
3.5.3 What about multiports? . . . 136
3.6 Multirules alone do not give concurrency . . . 137
3.7 Comparing multiport and multiwire concurrency . . . 139

4 Multiport Combinators **147**
4.1 Special decomposition of nets . . . 147
4.2 Combinators for multiport interaction nets . . . 153
4.2.1 The system . . . 153
4.2.2 Multiplexors and transpositors . . . 157
4.2.3 Menus and selectors . . . 161
4.2.4 Allocator . . . 163
4.3 Encoding (restricted case) . . . 163
4.4 Encoding systems with recursion . . . 172
4.4.1 Duplication . . . 172
4.4.2 Codes, copiers and decoder . . . 173
4.4.3 The encoding, general case . . . 175
4.4.4 Correctness of the encoding . . . 180
4.5 Quality of the combinators . . . 181

References **185**

Introduction

Same causes lead to the same effects. Modern science can only begin its existence with this premise on the uniformity of nature. Science presupposes determinism as a theoretical hypothesis – from the Laws of Newton to Laplace's demon – and as a working hypothesis, obtaining much success. With the discovery of quantum mechanics, interpretations of the world with non-deterministic are made. We are not entering this aporetic debate in this work. At best, we can say it inspired it.

In this thesis, we study a world in which we suppose that non-deterministic relations can exist. The main questions that we ask ourselves about such a world are how to define in them *equality* or *similarity*. If the same causes can have many consequences, how can we say two states of the world are similar if what follows from them is not well defined? Is it possible to think the equality of states of the world retrospectively, by the events they generate? To model such worlds, we need cause/effect relations, actions, events,... We pick the rewriting paradigm, strictly syntactical, where objects are *words* on a given alphabet and events are given by instantiations of a *rewriting mechanism* on parts of words. If it is debatable whether it is an acceptable model of the physical world or not, it is the most general approach to any type of calculus. Any computation is, syntactically, more or less complex rewriting mechanisms on more or less complex objects.

The formal study of computation is related to the formal study of logic, historically and technically, which reflected onto the methodology. The same questions were raised; equivalent answers were given. The former were: What can we prove? vs. What can we compute? Is this logical system equivalent to this one? vs. Are these two models of computation the same? Is this logic complete? vs. Is this model complete? What does complete mean? Relatively to what? Are there infinitely many logics/computational classes? The answers were extensional: this can prove this set of theorems. vs. This can compute this class of functions. Completeness: It can prove *all* theorems/compute *all* functions. What does it mean? This similitude was crowned by one of the most important results in the field – of logic or computer science, depending on which side you pick as secondary – that gave a *mathematical* reason to this symbiotic relationship, a *demonstration* that they were strongly related. Curry and Howard's discovery of the relation between λ-calculus and natural deduction [11, 30]. To every proof corresponds a program. The two worlds are for ever united. Executing

a program is like transforming a proof. Into what? Into an equivalent proof, in some sense a "simpler" one, a little like computing an arithmetical expression is transforming it into a simpler one, until the simplest one is obtained.

This correspondence is therefore possible thanks to a special point of view on proofs. Originally, formal logic was kind of static. A proof is this series of formulas which sequence is determined by some causal relation given mainly by the *modus ponens law*, and which proves the last formula of the sequence. In [25], Gentzen modifies radically the point of view on the matter. He concentrates on deduction laws more than on theorems, and most of all, introduces dynamic in the static world of formal logic. In his paradigm, proofs can be *transformed* into equivalent proofs, by a strict and rather complex rewriting mechanism. This opens the breach to the questions of similarity, finite/infinite transformations, measure of a proof,etc. The *proof* is the object of the study. In the setting, theorems are just a way of expressing that some of these objects have a meaning, that they are *correct*. As for the others... it stays unclear at first.

Many programs nevertheless do not correspond to proofs. A proof has a *result*, the theorem it proves. Computations on the other hand, do not necessarily end. Famous examples are non-terminating Turing machines or the λ-term $(\lambda x.xx)(\lambda x.xx)$. They do not have a result. So how can we compare them? Should all non-terminating programs be considered equivalent? Is a program that calculates indefinitely the digits of π, one at the time, equivalent to one that indefinitely outputs "Hello World", "Hello World", "Hello World", ...? Infinitesimal calculus can be applied here to answer negatively. The first program approaches π: that is its result. The second one approaches "Hello World" as a constant function: that is its result. But then, printing this sentence infinitely many times is the same as printing it once? Answers similar to the ones given by mathematics can be given in some cases. But not always.

For once, technology preceded the theory by far. Our computers are machines executing computations. But as a whole, a computer cannot be considered to compute "something". First, it can do several tasks at once, so it may not only have one, but *several results*. Second, its result depend highly on what is asked from them. It is *interactive*. The external user will determine the computations that will take place inside the machine. Its results, if ever defined, depend highly on its environment. How can we compare such open, result-free systems? If we cannot compare *what* they do, maybe the real question is: *how* do they do what they do? Because such a question is too wide, we restrict ourselves to its relativization: do these systems do whatever they do in the *the same way*?

Interaction nets In this work, we chose to study a graphical model of computation, that has the advantages of being *multitask* – *i.e.* of parallel computing – and *open* – objects can interact with the environment: *Lafont interaction nets*. Lafont interaction nets, presented first in [32], are graphical computation languages in which the minimal

step of computation is given by rewriting rules on pairs of *cells*. This way, a program, having a cell a but missing some cell b can interact with another program which has b, thus modeling interaction (as often in computer science, we do not differentiate the programs from their environment, which we consider to be another program). More than a calculus, it is primarily a programming paradigm; it is left to the user to define the primitive objects to use and the rewriting rules over these objects. Its great freedom is clearly one of its advantages. It gives a simple and really powerful way of programming a large panel of algorithms.

Primitives are *labeled cells* with a definite amount of *ports* that depend on the label. *Wires* can connect two ports. Each cell has a special port, deemed *principal*. Only pairs of cells joined by their respective principal ports can interact. Such pairs are the calculus' redexes. Each redex can be rewritten by a rule totally defined by the labels of the involved cells. The *rules* are given *with* the system. At most one rule can be given per redex.

Lafont himself "programmed" arithmetics, Turing machines and other computational models. In his PhD thesis [37], Sylvain Lippi programs some famous algorithms as the Hanoi towers and several sorting algorithms. He moreover implements a programming language based on interaction nets. One can anyhow think of any kind of situation, build complicated rules from "complex" primitives. Hence, one can describe behaviors at different levels, scaling the language as he better sees fit, creating "macros" for instance, or taking as primitives to the language more or less complicated steps of computation. The only limitation is in the global expressivity: it allows only to express parallel but *deterministic* computations. Several computation steps can happens at once, but the result is always unique, like a Parallel Turing Machine, that can have a reading head on each of its tapes.

Since programs and environments are identified, and we wish to model non predictable environments, we model programs themselves as non-deterministic. The definition of interaction nets is restrictive. Several conditions can be readily relaxed, among which we retain three, yielding so called *multiruled, multiwire* and *multiport* interaction nets. They all allow non-deterministic behaviors.

These extensions have appeared naturally in the literature. Differential Interaction Nets of Ehrhard and Regnier [17] use several rules for some of the redexes. They could be a starting point for a Curry-Howard style equivalence between concurrent systems and logic. Concurrent nets of Beffara and Maurel [5] use wires that connect more than two ports, like names connect several prefixes in name-passing calculi. Yoshida also uses such connectors in her work on concurrent combinators [59]. Mazza encodes the π-calculus into interaction nets which cells can interact on more than one port. Such systems are studies for themselves by Banach [3] who defines them as a particular case of hypergraphs and studies their mechanism of deadlock creation. Fernàndez and Khalil [20] study a very important such cell and its expressivity: a cell that naturally represents resource management. Even though such applied results have discretely

flourished, a systematic study of such extensions has only been carried out, to our knowledge, by Alexiev in his PhD thesis [1].

He compares the different extensions by encoding them into one another. This means that for any system of one kind, he is able to define a system of another kind that has the same "behavior, without really defining what it means. As he says himself:

> "[W]e don't prove formally the faithfulness of our translations, but we introduce them gradually and give comprehensive examples, so we hope that we have made their faithfulness believable" ([1], p. 64).

He is aware of equivalences such as bisimulation for Milner's π-calculus, but such tools for interaction nets are not available to him yet. Moreover, he does not discuss at all the problem of divergence, which we believe is quite important. Because many of his encodings are based on commitment, he has to introduce *un-commitment* steps, which lead to cyclic reductions.

The question of relative expressivity of different concurrent extensions is quite important. Although all these extensions have been shown to be as expressive as π-calculus, the encodings are so different that the relations between them is yet unclear. The correctness of these encodings is proved with such *ad-hoc* techniques that a satisfactory comparison of these extensions is lacking. For instance, multiwires easily express the usual parallel operator of process calculi that connects names without bounding them, whereas multiports really seem fitted to emphasize π-calculus' sum. Non-interleaving semantics make us believe that these constructs are not completely equivalent. It would therefore be possible that that multiwires and multiports do not have exactly the same expressivity. We show on the other hand that multirules add no expressivity to the other two extensions. It says something about "coin-tossing" non-determinism, and the fact that it is not useful in the presence of connectors and multiport cells, so it also explains in some sense that non-deterministic Turing machines are not good models of concurrency. It also brings attention on Ehrhard and Laurent's encoding of π-calculus in differential interaction nets [16] which are a particular instance of multirule nets. Which of the two results is more accurate: the separation we propose or their encoding, since there is something more in multiport cells than in differential interaction nets.

Another problem is the choice of a "canonical" extension to model concurrency. Since they are all comparable with π-calculus (and other process calculi), such a choice is not obvious. An easy answer would be the most general one. But then, it lacks some simplicity. Why use multirules if they bring no expressive power? We can already say that the main result we obtain is that multiport nets are the good choice. They can perfectly encode any interaction net system. Most of all, it is the uselessness of connectors that is interesting in this result, since they seem a natural way of expressing the multiplicity of occurrences of names in processes. Getting rid of multiple connections mean that all interaction can be done privately, as soon as primitives can have several

connection points. This reminds of *internal mobility*, where all communication is made on bound names and which Sangiorgi proves to contain most of the expressive power of π-calculus [53]. The result is made possible in interaction nets by our decision to concentrate on *ports* as primitives for channels.

Finally, a more theoretical motivation of comparing different concurrent systems is the hope of constructing some hierarchy of non-determinisms. This notion is negative by definition – not deterministic. We might gain some insight on it if we can deal with some kind of "non-deterministic complexity". Before we can give a quantity to this complexity, we want to see if it has any sense. For that, one needs to be sure that there are non-equivalent systems; that some are actually more *powerful* that others, more complex. Here, we are more interested in separating families of interaction net systems. Separating here means to find some systems of one kind that cannot be expressed by any system of another kind. Here again, results depend on the choices that are made to compare expressivity, so a clear definition of equivalence is necessary.

We explore several ways to compare expressivity of concurrent interaction nets extensions.

- The first is purely semantical, based on *event structures*. An event structure describes the possible computations a net in a given interaction net system can perform. Comparing systems then amounts to comparing the event structures of their nets. A part of the job has already been done in [40]. We prospect it because of its elegance and because it hints on the other relative expressivity results.
- We then look for an *operational semantics* for interaction nets, simple enough to use. For this, we consider nets as hyper-graph rewriting systems for which a wide literature exists. We particularly look to define labels on transitions in a manner that allows us to construct interaction as synchronization of transitions. This is a usual approach of process calculi. Bisimulation can then be defined in a natural way. It anyhow reveals itself to be too complicated to be used for relative expressivity as the labels highly depend on the alphabet of the system.
- Finally, we define an *observational equivalence* based on a general notion of *barbs*. This really seems to be the right approach. It allows to compare nets of different systems, and therefore is fit for a good definition of translation.

It is only once we hold a good equivalence that we can consider defining good encodings. We avoid entering the debate on what a good encoding is by using one of the most exhaustive works on the subject, Gorla's paper on encodings [26].

To sum-up, in this thesis, we pursue Alexiev's study by making it more precise and more technically satisfactory. We also link it to other areas of computer science like graph rewriting and process calculi, of which we both use extremely powerful tools and to which we bring in exchange new results. One of our main accomplishments is to provide interaction nets with a good behavioral equivalence. For a more precise

definition of these equivalences, we define a textual language that *is exactly* interaction nets. In this language, all ports, even bound, are explicitly designated, making them the atomic particles of communication channels.

From channels to ports *Names*, representing communication channels, are one of the most effective ideas of process calculi for the purpose of expressing concurrency. They make connection independent from contiguity. As Honda remarks in [29], names are closely related to the non-determinism of concurrent computation in two ways: "First, names can represent sharing of interaction points [...] Secondly names serve for keeping identity in spite of possible change of meaning during computation". Honda goes on to show a formal equivalence of name-based calculi with what he calls *nameless processes* which have a graph-rewriting look and make use of a notion of *explicit connection*. Honda's nameless processes are really not far from interaction nets, multiport and multiwires allowed, while explicit connections are wires. It therefore seems that the underlying structure of name based calculi, the names, is very naturally represented by interaction nets.

On the other hand, *ports* appear naturally in graphical representations of concurrent calculi, such as Milner's π-nets [41] and Danos and Laneve's *graphical κ-calculus* [12], in which ports are called *sites*. Parrow actually uses the word "port" and insists on their importance in his *algebraic language for networks* [46]. However, in these last two examples reduction is not implemented by rendez-vous communication: it depends on a notion of state and ports are rather indicators of that state. As we have seen, ports are essential to interaction nets. They are the core of the connectivity and differentiate nets from hypergraphs where incidence is sufficient.

Our wish here is to go from a graphical representation of processes to an algebraic one, with the result of separating the two relations that names bear with non-determinism. While identity is maintained by identity of port names, sharing of interaction points is made explicit: ports all have different names but are bound together through explicit *connectors* that take the form of a finite set of ports. To clarify the relation between names and ports, one can think of ports to be different occurrences of a name, and a connector to be the explicit definition of which ports are occurrences of a same name.

Making the distinction of principal port and auxiliary port is what brought Lafont to his encodability result. Interaction nets are powerful thanks to that feature: for the same two cells to be able to interact or not while connected is a strong property that hypergraph rewriting cannot provide. Even though the two worlds seem so close, it is actually rather complicated to express interaction nets in the pure language of hypergraphs, as shows the detailed work of Banach [3]. Actually, it seems any attempt to define interaction nets in a more formal way than Lafont stumbles on the value to give to ports. The fact that ports are connection points that are limited to connecting

two elements is unnatural in any other setting[1].

The best approach for formalization (for one that considers Lafont's definition as not formal enough) seems to be *textual*. Giving a grammatical language for anything is always a must of formalization. An algebraic language for Lafont nets was given quite quickly ... by himself. Immediately in fact [32]. But a systematic study of the language was only started in [21], by Fernández and Mackie. That language is rather aesthetic as it relies on the tree decomposition of Lafont nets. Thanks to this, terms have a nice form, with a clear subterm relation, clear interface definition and some kind of leveling that is close to π-calculus prefixing. The nice tree structure of Lafont nets is lost once considered their concurrent extensions. This means that the textual language cannot be transposed directly to express these later, especially connectors.

Textual interaction nets We introduce a simple and flexible algebraic framework for concurrent interaction nets systems (INS). Instead of taking the inspiration in term rewriting, we follow what are the most common algebraic languages for concurrent systems – *process calculi*. In a way, we generalize the replication-free fragment of Laneve and Victor's solos calculus [35] (a connection between solos and interaction nets was already pointed out in [34]).

The basic components of interaction nets are *agents* – or *cells* – and *connectors* – or *wires*. An agent is an expression of the form $\alpha(x_1, \dots, x_n)$, where α is an arbitrary *symbol* and $x_1, \dots, x_n$ are ports, whose syntactic status is similar to names of usual name-passing calculi. Each symbol comes with a fixed *degree* which determines the number of ports of the cell, in this case n. A *k-connector* is a finite multiset of ports of cardinality k, denoted by $[x_1, \dots, x_k]$ (the use of multisets is actually a technicality; the reader may think of a connector as a set).

A *net* is a finite multiset of agents and connectors, *in which every port appears at most twice*. This restriction, which may seem unusual if one likens ports to names, is in reality absolutely natural: a port only has two "sides". If a net uses both of them, the port is *bound* (and may be renamed at whim, by α-equivalence); otherwise, it is *free*, and is an interface to the external world.

Given two nets μ, ν, one may suppose their bound ports to be disjoint, and consider their standard multiset union, which we denote by $\mu \mid \nu$. This operation may bind some ports, allowing the interaction between μ and ν. Two agents may interact when there is a connector between them: the net $\alpha(x_1, \dots, x_m) \mid \beta(y_1, \dots, y_n)$, under the hypothesis $x_i = y_j$, is called a *pair* and may be transformed into $\nu \mid [\tilde{z}]$, where ν is any net whose form depends on the tuple (α, i, β, j) and whose free ports are exactly $x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_m,\ y_1, \dots, y_{j-1}, y_{j+1}, \dots, y_n$. Therefore, interaction is binary. Several rules can be defined for an active pair, so tuple defining a rule are of the form

[1] Tom Hirschowitz has provided a profoundly formal definition of multiport interaction nets in terms of presheaves. Theoretically nice, it feels a bit overwhelming to deal with such complicated categorical concepts for something that is so simple to visualize.

(α, i, β, j, k) where k is an identifier of the rule for that pair. If at least one rule is defined, the pair is deemed *active*.

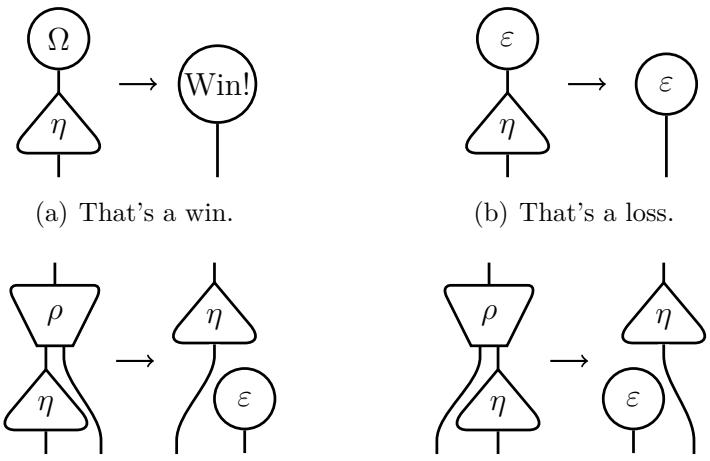
An interaction net system is defined by choosing a set of symbols (an *alphabet*) and by specifying a set of *interaction rules*, attaching a fixed net ν as above to a given tuple (α, i, β, j, k). Of course, not all tuples need to have an associated rule.

Formalization of interaction nets into a textual form brings a new point of view on the nature of the three non-deterministic extensions. The alphabet decides for the use of multiports or not while the set of rules gives ambiguity or not. Connectivity, on the other hand, is a *structural component* of interaction net systems. A given system can be multiwire or not, depending if one allows k-connectors or not, for $k \neq 2$.

Example Let us give an example. We consider 5 symbols, $\Omega, \eta, \varepsilon, \rho$ and Win!, of degree $1, 2, 1, 3$ and 1, respectively. There are 4 interaction rules given graphically in Figure 1 and textually by the following oriented equations:

$$\begin{aligned} \Omega(a) \mid \eta(a,x) &\to \text{Win!}(x) \\ \rho(a,b,x) \mid \eta(a,y) &\to \eta(x,y) \mid \varepsilon(b) \end{aligned} \qquad \begin{aligned} \varepsilon(a) \mid \eta(a,x) &\to \varepsilon(x) \\ \rho(a,b,x) \mid \eta(b,y) &\to \eta(x,y) \mid \varepsilon(a) \end{aligned}$$

The system may be seen as an extremely simple resource management model: the net $\mu = \Omega(c) \mid \rho(a,b,c) \mid \eta(a,x) \mid \eta(b,y)$ represents two agents η competing to access a resource Ω. The agent ρ is the resource manager; it grants access to Ω to the agent interacting first. The reader can check that μ has two normal forms, $\text{Win!}(x) \mid \varepsilon(y)$ and $\varepsilon(x) \mid \text{Win!}(y)$, corresponding to the two possible outcomes of the competition. A similar effect may be obtained by using a 3-connector instead of an agent ρ: the net $\mu' = \Omega(c) \mid [a,b,c] \mid \eta(a,x) \mid \eta(b,y)$ has essentially the same behavior as μ.



(a) That's a win. (b) That's a loss.

(c) Fight! (d) Fight!

Figure 1: Rules for the resource management interaction net system.

A remarkable example is obtained by considering two families of symbols $(\iota_n)_{n\in\mathbb{N}}$,

$(o_n)_{n\in\mathbb{N}}$, with ι_n, o_n of degree $n+1$, with the rule $\iota_n(x, y_1, \ldots, y_n) \mid o_n(x, z_1, \ldots, z_n) \to [y_1, z_1] \mid \cdots \mid [y_n, z_n]$. If we write $x\tilde{y}$ for $\iota_n(x, \tilde{y})$ and $\overline{x}\tilde{y}$ for $o(x, \tilde{y})$, we see how this interaction net system is essentially the replication-free solos calculus, with explicit fusions. We hope that these two examples give the reader a glimpse of the versatility of interaction nets, and their ability to express equally well high-level and low-level models.

Overview of this work After introducing the needed definitions, the first chapter deals with the semantical approach to expressivity based on event structures. This gives us intuitions on relative expressivity, but lacks of a good notion of encoding. Chapter 2 is a study of interaction nets as hypergraphs and produces a structural operational semantics for them, but is not well fit for language comparison. It is only in the third chapter that we use the textual language for nets, as the observational equivalence we give, based on a notion of barb, is strongly inspired from process calculi equivalences. We are able then to define a good notion of translation of a system into another that allows us to give both positive encodability results and separation results. In this third chapter, we compare all combinations of concurrent interaction nets. The main result of this study is that multiport interaction nets are powerful enough to encode systems of any other kind. To emphasize this, we dedicate Chapter 4 to a multiport system capable of expressing any multiport interaction net system. This system is therefore universal for interaction nets in general.

Event structures In [40], Damiano Mazza proposes a very formal approach to study relative expressivity of rewriting systems. He studies interaction nets through the lens of Winskel's *event structures* [58]. The assumption is that a computational process can be described as a collection of *events* which are related by *causality* and *conflict*, very much in the spirit of Poincaré's definition of time [48]. An event f is caused by an event e if f cannot occur until e has first. If e and f are not in a causal relation, then they can occur independently and one can speak of parallelism. On the other hand, events e and f are in conflict if the occurrence of one annihilates the possibility of the other to occur. The expressivity of a rewrite system in this setting is in some sense given by the complexity of its causal and conflict relations. For instance, the causality and conflict of Turing machines is trivial: each transition depends on the preceding one, while it is more complex for π-calculus.

Of course, defining an absolute complexity measure of event structures seems rather insignificant, so one prefers to define some kind of relative complexity. Based on *history-preserving bisimulation* on event structures, introduced independently by Rabinovitch and Traktenbrot [49] and van Glabeek and Goltz [57], Mazza introduces the notion of *bisimilar embedding*. Intuitively, en event structure E embeds in an event structure F if F contains a substructure E' bisimilar to E. It means the system underlying F is able to yield a structure at least as complex as the one underlying E.

It is rather straightforward to determine the event structures corresponding to some interaction net, and therefore the class of event structures that actually characterizes a family of interaction nets. Using a local property on event structures – *confusion* – Mazza gives a first abstract separation result: event structures of systems with confusion cannot be embedded in the event structures of systems without confusion. A straight application is that differential nets are strictly less expressive, as event structures, than usual process calculi in which confusion is determinant. For us, the immediate consequence is that multiwire and multiport interaction nets are strictly more expressive than multirule interaction nets. We apply this technique to show that some structures of multiport nets cannot be embedded in event structures of multiwire nets without introducing divergence. Even though these results give us a certain idea of the hierarchy of expressivity of different extensions of interaction nets, they suffer from some strict properties of event structures. Especially, embeddings of event structures are not related to the possibility of encoding, since they do not take into account any syntactical constructs of the studied systems.

Operational semantics The second methodology of comparison is based on labeled transition systems and bisimulation. It is a standard technique in process calculi as π-calculus and even the more basic calculus of communicating systems (CCS). In these calculi, synchronization is decomposed in atomic actions that are usually of the kind "send" or "receive". A process can evolve by performing one of the above. The channel on which the message is sent or received, along with the message itself and the type of the action form the label of the transition. In interaction nets, this approach can be applied in the following way. If the net contains a cell of type α which principal port is free, it can perform any action α is able to perform. This is due to the locality of interaction. Depending on the active pairs α can be part of, several such actions can occur. Can all these actions be labeled by α and the port name? The label needs to contain information on the "opponent" of α. Conversely, is the single information of the opponent enough to label the transition?

Moreover, in process calculi, the labeled transitions semantics is often given in a *structural* form: two processes than can perform dual actions can be combined into a process performing an internal reaction. This compositionality allows a very elegant definition of the semantics, that really takes into account the local nature of interaction. In interaction nets, this means that if there is a rule for a pair α/β, a net containing an α with free principal port can be combined with a net containing a β with free principal port and yield an interaction. Compositionality should therefore be given on labeled transitions. It is not straightforward to define the equivalent of the *communication rule* of CCS in interaction nets. The fact that a cell can interact with several other cells, or in several ways, makes it hard to be sure that the transitions are compatible, even if the combination of the two nets clearly creates an active pair.

A fruitful field in which labeled transition systems for *non-standard* calculi are

defined is the area of graph transformation, of which interaction nets are a particular case. Such semantics have been developed from "reaction rules" [55, 19] and the ambients calculus [51, 7]. An interesting fact about these semantics is the context independence of the resulting behavioral equivalences, *i.e.* they are congruences.

In graph transformation calculi, labels are considered to be the "minimal context" in which a reaction can take place. This approach is dual to the one of process calculi. Here, the label says: if some process that is able to receive a message is connected to this graph, this latter is able to perform an action. The idea of minimal context has been formalized by Leifner and Milner [36]. We adopt an equivalent approach more suitable for the case of hypergraphs: the Borrowed Contexts technique [19]. It is a categorical approach based on the definition of rewriting rules as double pushouts. By a "smart" diagrammatic construction, a minimal context for reaction is found.

This approach is anyhow "monolithic". The diagram can only be constructed on the whole hypergraph, making it impossible to define an equivalent of the communication rule of CCS. We focus on the search of an equivalent of the communication rule, as it yields a nicer semantics overall. It seems natural too, since interaction is extremely local in nets. The general direction here is towards formal results that support the slogan that graph transformation and process calculi have essentially the same descriptive power, as witnessed by a large variety of graphical encodings for process calculi. This slogan is well established for reduction semantics of closed systems and calls for an extension to open systems. The potential advantage of graph transformation over process calculi is their inherent generality, as one will seldom study a particular graph transformation system for its own sake: the results hold for graph transformation systems in general. Since we are interested in the paradigm of interaction nets, it seems a good point of view.

However, this technique ends up being too general for our purpose, and at the same time leads to a too restrictive semantics. The purpose is not completely achieved as the communication rule cannot combine transitions from transitions only; some extra information is always necessary, which finally amounts to the use of the full interaction rule. We give some ways to bypass that problem in the case of simply wired nets, which is still not sufficient, but gives us yet another reason to believe simple wires are the way to go for interaction nets.

Observational equivalence In the third chapter, we study a third behavior in a third way, by looking for an observational equivalence. A testing equivalence may work within a given system, but it behaves poorly with respect to encodings. As Parrow observes [47], it is not fit for a full abstraction result: the contexts in the encoding language might be of greater number than in the encoded one, even if they do not surface in the encoding, and can separate the translations of two equivalent terms. The barbed equivalence approach is more satisfactory, but defining $\mu\downarrow_x$ (the net μ has a barb on x) regardless of the system seems hopeless (the alphabet and interaction

rules may be virtually *anything*). Abstract approaches such as the one of [50] do not work satisfactorily for interaction nets, mainly because of the possibility that symbols interact with themselves (so a symbol would belong to its own orthogonal, which defies the definition).

Our solution is to combine the *may-testing* approach and barbs, using the first to define the second. We assume that some minimal information concerning the observable behavior of nets is provided with the system itself. After all, in name-passing calculi, it is usual to consider as a barb a name on which a process can interact in a significant manner. In interaction nets, an interaction can be pretty much anything, so we let the author of the interaction net system define for herself/himself which computations are significant. Therefore, the complete definition of an interaction net system is a triple: an alphabet, a set of interaction rules, and a subset of *observable rules*. We then write $\mu{\downarrow_x}$ if there exists a net o, called *observer*, such that $\mathrm{fp}(\mu)\cap\mathrm{fp}(o) = \{x\}$ and such that $\mu \mid o$ generates an observable computation. Furthermore, we must require that, in $\mu \mid o$, such a sequence truly comes from the interaction of μ and o and is not already present in μ or o alone. Once barbs are given, *barbed bisimilarity* and *barbed congruence* are obtained in the standard way.

Taking parts of the interface as observables is natural in process calculi. We have seen that labeling transitions is problematic since the labels depend highly on the actual language, while ports are an atomic part of all interaction nets. This way, we can compare observability on different languages, which is our purpose.

Expressivity We then proceed to introduce the notion of *translation*. It is based on an almost straightforward reformulation, in interaction nets, of fairly standard properties which are asked of encodings between process algebras. We take as main reference Gorla's work [26], whose thorough analysis of the literature on encoding and separation results approaches exhaustiveness.

A translation builds for any system $\mathcal{S}$ of one family of IN systems, a system $\mathcal{T}$ of another family, such that, for every net μ of $\mathcal{S}$, there is a net ν of $\mathcal{T}$ which has the same behavior as μ. In [26], Gorla gives a list of five criteria which a *valid* encoding between concurrent calculi should meet, distilled from the common properties on which the existing literature (which is quite vast) seems to converge:

i. Compositionality
ii. Name invariance
iii. Operational correspondence
iv. Success sensitiveness
v. Divergence reflexion

The first ensures the preservation of the degree of distribution, the fact that a *decider* is not introduced to simulate locally taken decisions. The second guarantees that all ports have the same meaning: the encoding cannot depend on the names of the ports. Operational correspondence is the weakest form of equivalence that can

be asked of an encoding. It says that a net and its encoding have in fact the same behavior. Success sensitiveness, which takes a particular form in Gorla's work because he considers mostly languages with a prefix constructor, excludes trivial encodings, which are not banned by the other conditions. We use instead, with the same scope, a *bisimilarity condition*, which stipulates, most of all, that a net and its encoding have the same barbs. The last condition about divergence is the most debated one. We follow Gorla's intuition and believe that an encoding that necessarily introduces divergence is weaker than one that does not. A separation based on that criteria alone is weaker than a proof of the non-existence of an encoding all together, but the least one can do is to make sure to know the dangerous patterns that can lead to divergent encodings.

The case of interaction nets To test the sensibility of our definition of behavioral equivalence and to investigate the expressivity of interaction nets, we propose an encoding of the π-calculus in a suitable system (which is unrelated to the encoding of [38]). It is interesting because it shows how replication, a highly non-local operation in process calculi, may be implemented by means of completely local rewriting rules in interaction nets.

We then proceed to the core of the work: internal expressivity results. We compare all non-deterministic versions of interaction nets and their combinations by means of translatability. This results in a non-strict hierarchy of systems. In synthesis, we can say that multirules, *i.e.* coin tossing, can be considered as the weakest form of non-determinism. It is always possible to get rid of them in the presence of multiwires and/or multiports. Conversely, they are not able, on their own, to encode k-connectors or multiport cells.

$$\text{MWR} \sim \text{MW}, \quad \text{MPR} \sim \text{MP}, \quad \text{MPWR} \sim \text{MPW}, \quad \text{but} \quad \text{MR} < \text{MW}, \text{MP}.$$

The key to this result is that non-determinism in a simple net with multirules is in some sense internal. Whatever happens inside such a net, an observable port will for ever remains such. It is not at all the case in the presence of connectors and multiport cells.

On the other end, multiport cells are omnipotent: k-connectors ($k \neq 2$) can be eliminated in the presence of multiport cells. Since multiple rules can also be gotten rid of, to any INS can be associated a strictly multiport system completely simulating it. The converse relation is more delicate. In the absence of reflexive rules (see below), it is not possible to encode multiport cells into a uniport system without introducing divergence (the separation is weak, leading us to the use of a non-strict order between the two systems):

$$\text{MPW} \sim \text{MP} \quad \text{and} \quad \text{MW} \leq \text{MP}.$$

As a result, accepting or not this weak separation result, processes in which each

name is shared by exactly two peers are sufficient for modeling concurrent computation. This result, of which we saw an instance in the resource management example, in which a 3-connector is replaced by the agent ρ, may be seen as a nice formalization of a phenomenon remarked by Danos and Laneve [12]: "That such arbitrary transactions are reducible to peer-to-peer interactions is a fact which should be, but for some reason is not, a classical result in the theory of the π-calculus".

Universality We have build a full hierarchy of interaction net families. In the last chapter, we build a particular INS, capable of expressing all the others. As Lafont himself puts it in [33]:

> By definition, a *universal interaction system* has the property that any other system can be translated into it.

Simon Gay [24] follows closely the mechanisms of combinatory logic to define combinators for Lafont nets, composed of 8 symbols. Lafont takes a step aside and combines several actions of Gay's combinators in order to obtain a smaller system, composed of three cells, ε, δ and γ, and all interactions for them, thus 6 rules. The rules are nevertheless very simple, unlike Bechet's minimal universal system with 2 kinds of cells [4]. He combines Lafont's δ and γ in one cell, but the rules that come out of this process are ... to complicated to be understood.

We wish to define a universal system for concurrent nets. This has a double purpose. First, it would give interaction nets a natural system that can be studied and dissected for itself, using techniques like *geometry of interaction* following the steps of De Falco [13] or programming an actual language, extending Lippi's `inn`. But it can also help us understand the minimal laws of computation for concurrent systems. Some are widely accepted, like the parallelism or name restriction. Others are discussed, as prefixing, which does not seem primordial taken all existing solo-style languages, or the sum representing choice. In Section 3.2, we show interaction nets code the π-calculus, so they can be considered a good enough model of concurrent calculi. If we are able to give an INS, universal but simple enough to have usable rules, one can consider these steps to represent the basic operations of concurrent languages.

We prove in Section 3.4 that simple connections are enough to encode all interaction nets. So unlike Yoshida [59] who studies *shared* combinators, meaning multiwire in interaction net dialect, we focus on multiport nets with simple rules and simple wires. In his PhD thesis [39], Mazza gives a universal system for multiport interaction nets. He only shows the universality in the case of systems which cells have at most 2 principal ports, but most of all, his definition of observability is hardly related to ours, so his combinators do not completely fit our purpose. Finally, we have more cells, but the rules are slightly simpler.

We give a universal system for multiport interaction nets with 6 cells and an infinite family, of which only a finite amount is needed to encode any given (finite) system. It contains Lafont's combinators and the encoding uses most of the constructions Lafont

uses in the case of simple nets. “News” cells are required for several reasons. First because the source cells are now multiport. Second, because not all active pairs yield observable interaction, so we use a particular cell for simulating observability. Third, not all cuts are active pairs so we introduce a blocking mechanism and its corresponding unblocking mechanism.

We partially achieve our goal. The rules are simple and interpretable and the cells not so many, even though we believe some improvements can be made.

Plan of the thesis

Chapter 1 Interaction nets

In this chapter, we introduce the paradigm of Interaction Nets, which will be studied in detail along the entire work. The first part is therefore devoted to definitions and simple results already present in Lafont's original works. His universal system is briefly discussed, as it is useful for the second chapter.

Section 1.1 is devoted to the description of some generalizations of interaction nets that can capture some non-deterministic behaviors. These generalization are created by relaxing some conditions that are put in by Lafont for his definition of nets, since he wished to model deterministic computations.

These extensions appear sparsely in the literature but have only been comparatively studied by Alexiev [1]. So the last part is devoted to already known *relative expressivity* results: whether the non-deterministic extensions can be encoded or not into one another. Section 1.3.1 summarizes the work of Alexiev. In 1.3.2, we recall some definitions around event structures and give some results of expressivity based on that technique taken from [40], in which the author uses these abstract constructions for comparing rewriting systems. We follow this method and add a separation result between two of the extensions in Section 1.3.3.

Chapter 2 Structural Operational Semantics for interaction nets

In the present chapter, we try to provide interaction nets with a proper operational semantics. In particular we describe a CCS-*like* labeled transition semantics for graph transformation systems.

We first discuss a straightforward out-of-the-blue definition of a SOS-semantic for simple interaction nets. The problems we meet make us try another approach, that of hypergraph rewriting, considering that interaction nets can be seen as particular hypergraphs. For this, we recall the basic definitions and concepts for the concrete case of hypergraphs in Section 2.2; in particular we give a brief review of the Borrowed Context technique. In Section 2.3, we provide a reformulation of the Borrowed Context technique in analogy to Milner's CCS; however, this analogy is imperfect as there is no need for a counterpart of the communication rule. This issue is addressed in Section 2.4, where we present our main results, which allow to define a graph transformation counterpart of a communication rule. These results are applied in Section 2.5 to obtain a satisfactory SOS like reformulation of the Borrowed Context technique. We conclude in Section 2.6 with some restrictions on hypergraphs and their rules that allow for a simplified, and sometimes usable definition of the communication rule de-

rived in previous sections in the framework of interaction nets. This definition allows us to formalize the initial intuitive SOS-semantics.

Chapter 3 Concurrent Interaction Nets

In this chapter, we study the expressivity of the concurrent extensions of interaction nets. A general definition of a labeled transition having partly failed, we consider here the commonly used approach of *encodability*. What it means exactly is described in Section 3.1.

We first deal with the general expressivity of interaction nets by giving a INS that encodes the π-calculus (Sec. 3.2). Finally, we proceed with the core of the chapter: comparing different concurrent extensions among themselves. Section 3.3 summarizes the encodability and separability results, that are then given in detail: 3.4 show how to encode any interaction net system in a multiport one; 3.5 shows how to express rule ambiguity with multiwires; in 3.6, we show that multirules alone are not enough to replace multiwires; finally, a weak separation result of multiport from multiwires is given in 3.7.

Chapter 4 Multiport Combinators

In this last chapter, we introduce a universal system for strictly multiport interaction nets, *i.e.* simply wired with simple rules. Since we have proven in the previous chapter that any interaction net system can be encoded into one of these, the *multiport combinators* we present can be considered as a universal system for general interaction nets.

The procedure relies on the ability to decompose a net, namely the right-hand-side of a rule, into two nets which have no active pairs and no deadlocks, making it possible to erase and most of all duplicate them. A splitting of this sort is discussed in Section 4.1. Section 4.2 presents the language of multiport combinators and some constructions that allow to encode system without recursion. Such an encoding is given in Section 4.3. The general case of encoding system with recursion is detailed in Section 4.4, after describing the duplication procedure (4.4.1). A last short section (4.5) question the quality of the system.

Chapter 1

Interaction Nets

In this chapter, we introduce the paradigm of Interaction Nets, which will be studied in detail along the entire work. The first part is therefore devoted to definitions and simple results already present in Lafont's original works. His universal system is briefly discussed, as it is useful for the second chapter.

Section 1.1 is devoted to the description of some generalizations of interaction nets that can capture some non-deterministic behaviors. These generalization are created by relaxing some conditions that are put in by Lafont for his definition of nets, since he wished to model deterministic computations.

These extensions appear sparsely in the literature but have only been comparatively studied by Alexiev [1]. So the last part is devoted to already known *relative expressivity* results: whether the non-deterministic extensions can be encoded or not into one another. Section 1.3.1 summarizes the work of Alexiev. In 1.3.2, we recall some definitions around event structures and give some results of expressivity based on that technique taken from [40], in which the author uses these abstract constructions for comparing rewriting systems. We follow this method and add a separation result between two of the extensions in Section 1.3.3.

The concepts for this chapter are introduced in the paragraphs about interaction nets (p. 2) and event structures (p. 9).

1.1 Lafont's calculus

1.1.1 Interaction nets

The language of Interaction Nets is one of graphical nature. The atomic pieces, *i.e.* the symbols, are *cells* – denoted by small Latin letters $a, b, c \ldots$ – which have a *label* – usually denoted by small Greek letters – $\alpha, \beta, \gamma \ldots$ – and a certain number of *ports* onto which can be connected *wires*. Each symbol has a determined number of ports depending on its label. We say a port *belongs to* or is *incident to* a cell. One of the

ports is more important than the others in a sense that will soon be made clear. It is referred to as the *principal port* while the others are called *auxiliary*. The number of auxiliary ports of a symbol defines its *arity*.

We represent cells like this: 

If $n = 0$, like this: 

Ports are not interchangeable and are implicitly numbered counterclockwise, starting at the principal one. The arity of the cell labeled α above on the left is n. If $n = 0$, then the cell has no auxiliary port and we represent it as a circle. Notice that a cell necessarily has a principal port.

A *net* is a graph like object composed of cells, ports and wires such that every port (belonging to a cell or not) is connected to a wire. Wires are each *connected* to zero or two ports exactly. We will sometimes say the wire *connects* two ports and that the ports *belong to* or are *incident to* a wire. Two wires connected to a same port are considered as one wire connecting the two other extremities of the two wires: the port in middle is forgotten about. The case of a 0-wire is called *cycle* and is represented by a circle.

A port can only be incident to one or two objects: a cell and a wire, a wire and a wire or just one wire. In the last case, the port is called *free port* of the net. The set of free ports of a net forms its *interface*.

In short, a net is an undirected graph with labeled incidence-ordered vertices, with the strange feature of having pending edges. An example of a net is given in Figure 1.1.

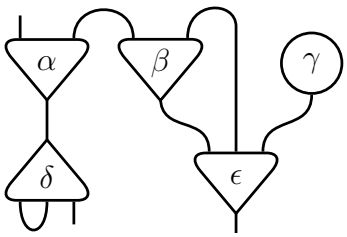
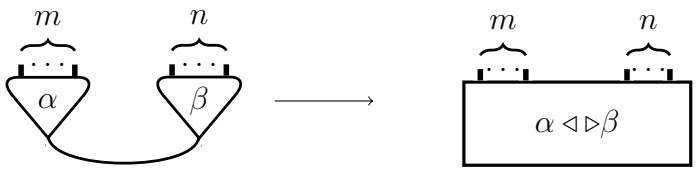
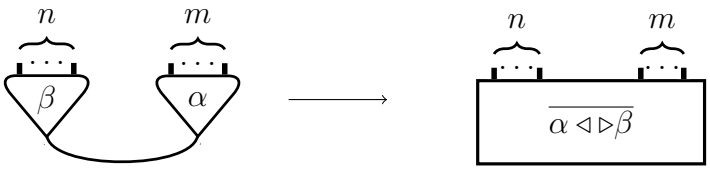


Figure 1.1: An example of a net (where ports are not shown, but implicit from the wiring)

Rewriting rules are defined for pairs of labels. Two cells which labels appear in a same rule and that are connected by their principal port can interact. Such a configuration is called an *active pair* (in general rewriting vocabulary: *redex*) and denoted by $A \bowtie B$. We represent a rule for $\alpha \bowtie \beta$, where α has arity m and β has arity n as follows:



where $\alpha \triangleleft \triangleright \beta$ is a net with $n+m$ free ports in its interface and is called *right-hand side* (RHS for short) of the rule for $\alpha \bowtie \beta$, which in return is called *left-hand side* (LHS) of the rule. A rule is then denoted $\alpha \bowtie \beta \to \alpha \triangleleft \triangleright \beta$. By abuse of language, the rule itself is sometimes referred to as $\alpha \bowtie \beta$. Of course, since nets are graph-like objects, they are subject to isomorphism, so the symmetric rule is immediately valid:



where $\overline{\alpha \triangleleft \triangleright \beta}$ is the symmetric image of $\alpha \triangleleft \triangleright \beta$. Be careful that the symmetric image of a cell is itself, meaning that ports are still given in counterclockwise order. The symmetry is on the graph-like object, not on the "drawing".

Some redex-like configurations do not have rules that can be applied to them. Following Lafont, we call *cut* any net composed of two cells connected by their principal ports. It is *reducible* if it is an active pair and *irreducible* otherwise. Irreducible cuts lock the computation: it is impossible to get rid of them (or duplicate) as they are nets in which all principal ports are bound but do not trigger interactions. Such nets are called *deadlocks*, and a net is said to be *deadlock-free* if none of its subnets is a deadlock.

The computation goes as follows. Given a net $\mathcal{N}$ and two cells c, d with respective labels α, β for which there is a rule and such that their principal ports are connected in $\mathcal{N}$, the two cells are removed and the hole replaced by $\alpha \triangleleft \triangleright \beta$ where the interface if connected to $\mathcal{N}$ following the rule's identification of ports. The configuration of the cells c, d connected by their principal port is called an *active pair*, and is denoted by analogy to the rule $c \bowtie d$. The rules have to be such that their is no possible ambiguity, *i.e.* if a rule is given for α, β, then their is no other rule for this pair and neither for β, α; and if a rule exists for α, α, then $\alpha \triangleleft \triangleright \alpha = \overline{\alpha \triangleleft \triangleright \alpha}$. A set of such rules is said to be *unambiguous*. Finally, right-hand-sides of rules have to be *reduced* , or *cut-free* (they do not contain active pairs) and *deadlock-free* (no part of the net is in some sense cut out of the computation).

The condition about possible redexes is justifiable by Proposition 1.1.2 ; if there were any active pairs, it suffices to reduce them inside the right-hand side of the rule to get rid of them, without changing at all the characteristics of the system. Except of course if there is a divergent subnet, which is not reducible to a normal form. Such

a possibility is anyhow undesirable. As are undesirable any deadlocks, since they are a piece of program that can neither be erased, duplicated nor interacted with. Why would anyone willingly put such a defect in a system?

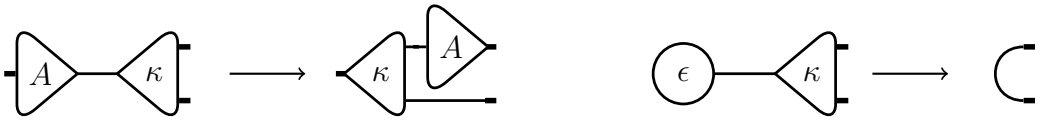
But more importantly, the technical consequence of violating these conditions would be that the translation into the universal system (actually, any translation in any system) would not enjoy the strong property of translations: the encoding of a net necessarily reduces to the encoding of its reducts. The technical price to pay is rather low compared to the achieved results. Anyhow, if one works with equivalences based on simulations, translations do not need such a strong property to be considered correct, so the conditions on rules become obsolete. Conversely, one can wish to study deadlocks or divergent systems, so one can consider useful to allow them. In Lafont's interaction nets, it is quite easy to detect deadlocks, so the gain is worth the trouble.

Definition 1.1.1. An *interaction net system* (INS) is a pair $(\Sigma, \mathcal{R})$, where Σ is a set of labels and $\mathcal{R}$ an unambiguous set of rules on pairs of elements of Σ.

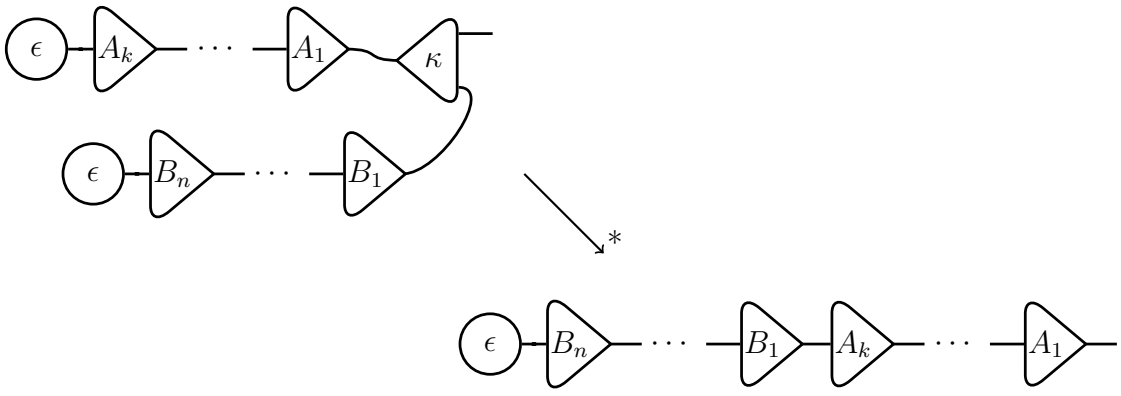
Let $\mathcal{S}$ be an interaction net system. Let $\rightarrow$ be a relation on $\mathcal{S} \times \mathcal{S} : (\mathcal{M}, \mathcal{N}) \in \rightarrow$ (usually denoted $\mathcal{M} \rightarrow \mathcal{N}$) iff $\mathcal{M}$ has an active pair $c \bowtie d$ and reduces using the rule for this active pair into $\mathcal{N}$. $\rightarrow^*$ is the reflexive transitive closures of $\rightarrow$.

We use various vocables for interaction. As you have noticed, we sometimes say *interaction*, and sometimes *reduction*, with all their derivatives. Normally, interaction will be reserved for one step of "reduction". We will sometimes say a net reduces some active pair, or interacts *through* it, but also that the active pair *triggers* the interaction.

Example : Lists We encode elements of a list by cells of arity 1 labeled by the value of the element of the list it represents; the end of the list is a cell ϵ of arity 0. The concatenation function uses a cell labeled κ of arity 2 with the following rules:

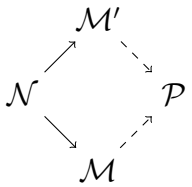


We let the reader verify that the computation goes as follows:



At each moment of the computation, only one active pair is present in the net. We can concatenate another list $C_1, \ldots, C_m$ through another κ-cell in the same way $B_1, \ldots, B_n$ is here. After the first step of reduction of the only active pair, two active pairs all simultaneously present in the net. The choice of which to reduce has no consequence on the final result, a list $A_1, \ldots, A_k, B_1, \ldots, B_n, C_1, \ldots, C_m$.

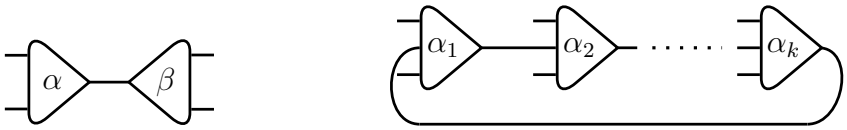
Proposition 1.1.2 (Strong Confluence)**.** *If a net $\mathcal{N}$ reduces in one step to $\mathcal{M}$ and $\mathcal{M}'$, then these latter both reduce in one step to a common net P.*



The proof is straightforward. It suffices to notice that rules do not overlap. Indeed, if $\mathcal{N}$ can reduce into $\mathcal{M}$ applying a rule r to a pair of cells c, d and into $\mathcal{M}'$ applying a rule r' to a pair of cells c', d', then c, c', d, B' are all different, so c', d' are still "present" in $\mathcal{M}$ and c, d are still "present" in $\mathcal{M}'$. It is therefore possible to apply r' to $\mathcal{M}$ and r to $\mathcal{M}'$ obtaining a same net $\mathcal{P}$. Notice that the proof does not make use of the restrictions on rules, so the confluence is always valid, and can be then used to justify the absence of redexes in right-hand sides of rules (in absence of infinite computations).

As mentioned before, it is rather easy to define deadlocks in interaction nets. They are important for the universality result which is based on duplicating and erasing nets. The problem with a deadlock is that it cannot be submitted to these two operations, that is why Lafont insists on their study in [32], where he even gives a class of nets which do not *introduce* deadlocks during computation. We will only need to check that some given nets do not contain them.

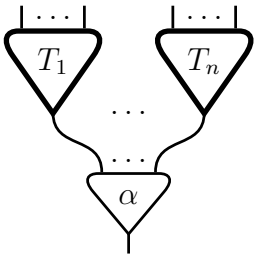
A *deadlock* is either a redex-like configuration between a pair of cells which labels do not form a rule (on the left) or a *vicious circle* (on the right):



The case where $k = 0$, *a.k.a.* the cycle, has already been mentioned. Forbidding deadlock in rules is not enough to completely avoid them. They can appear during computation; this is why we allow the 0-wire in the language.

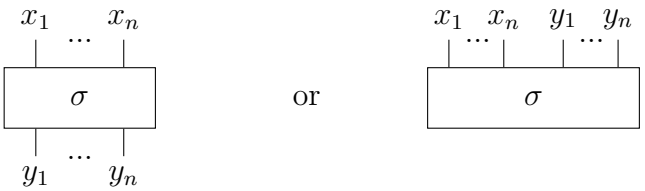
With this characterization of deadlocks, it becomes clear what a reduced net is: no redexes and no deadlocks. Let us continue by define some other particular nets.

Definition 1.1.3 (Tree)**.** A *tree* is a net with one distinguished free port called *root*. It is either a single wire, in which case the root is fixed arbitrarily, or it is obtained by plugging n auxiliary ports of a cell α into the roots of n smaller trees $T_1, \ldots, T_n$:

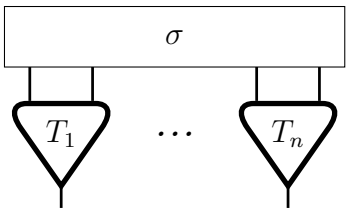


in which case, the root is the free port connected to the principal port of α.

A *wiring* is a net without cells and without cycles. So it is just a pairing of its free ports. A permutation σ of $\{1, \ldots, n\}$ defines a wiring with $2n$ free ports represented as follows:



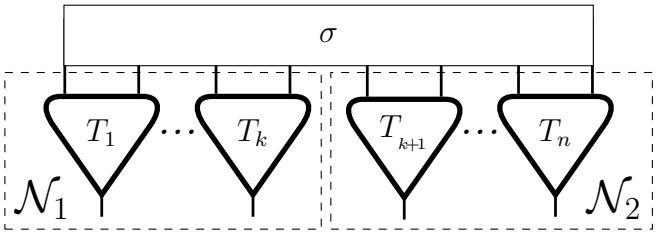
Proposition 1.1.4. *Any reduced net $\mathcal{N}$ with n free ports can be uniquely decomposed as follows:*



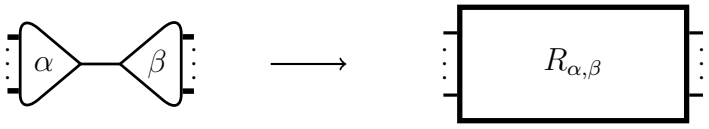
where $t_1, \ldots, t_n$ are trees (remember that a tree can be a wire) and σ a wiring.

We are now equipped with a good characterization of acceptable nets. Even though deadlocks can appear during computation, deadlock free nets enjoy a nice decomposition property:

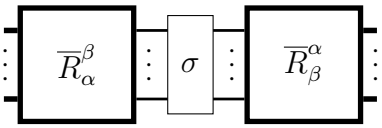
Proposition 1.1.5 (Splitting)**.** *Let $\mathcal{N}$ be a reduced net and $\tilde{z}_1, \tilde{z}_2$ a partition in two of its interface. Then there is a natural* splitting *of the whole net into two nets $\mathcal{N}_1$ and $\mathcal{N}_2$ s.t. $\tilde{z}_1$ (resp. $\tilde{z}_2$) belongs to $\mathcal{N}_1$ (resp. $\mathcal{N}_2$) and the two subnets are connected through a wiring.*



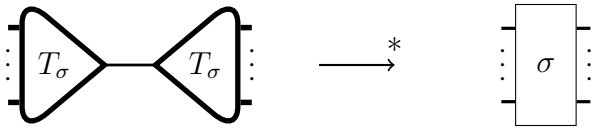
This is particularly useful to decompose right-hand sides of rules. Let us take a rule for (α, β):



We can now split $R_{\alpha,\beta}$ into two forests R^β_α and R^α_β, such that the ports connected to auxiliary ports of α are in the first net and those connected to β in the second:



Moreover, Lafont builds for any wiring σ a tree T_σ such that connecting two such trees by their principal port yields the desired wiring (for an implementation of that, see Section 4.2.2):



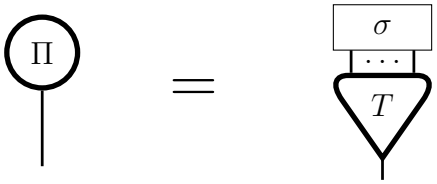
This tree can be integrated into each half $\overline{R}^\beta_\alpha$ and $\overline{R}^\alpha_\beta$, so the interaction can be decomposed (artificially, since the first interaction is not correct: its RHS contains active pairs):

One can informally interpret R^β_α as the net "hiding" in α and revealed by the interaction with β. This suggests a way of encoding a system into a universal system: a cell α shall be translated into a net containing in some sense all the hidden nets that can be revealed by recursive interaction. We will see such a translation with a little more details in Section 1.1.2. It is also useful to define an SOS-semantics as we shall see in 2.1.

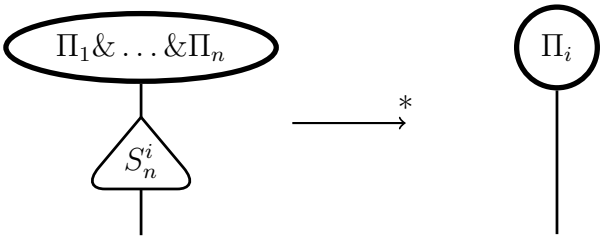
1.1.2 Universality

Lafont gives a simple interaction net system which enable to encode any interaction net system. Surprisingly enough, it contains only three labels and six rules; Another interesting feature, is that in the universal system, all pairs of cells can interact, even if it is not the case in the original one. We say this system is *full*.

To achieve this goal, Lafont identifies, along with trees we have seen before, a special kind of nets: a *package* is a reduced net with only one free port. Because a reduced net has no redexes and no deadlocks, a package has the following decomposition:



where T is a tree and σ a wiring. The shown structure of packages makes it easy to erase or duplicate them. It is also easy to construct a big package $\Pi_1 \& \ldots \& \Pi_n$, that Lafont calls a *menu*, composed of n smaller packages. He also encodes a *selector* S_n^i which can extract the i-th package of a menu of n packages:



We can already guess the idea behind the encoding. We denote by $[\alpha_i]$ the encoding of α_i and by $[\mathcal{N}]$ the net $\mathcal{N}$ where each cell has been replaced by its translation. Consider an interaction net system with p symbols $\alpha_1, \ldots, \alpha_p$. Then the encoding of a cell c labeled α_k will contain a menu $[R_{\alpha_1}^{\alpha_k}] \& \ldots \& [R_{\alpha_p}^{\alpha_k}]$ which represents all possible futures of cells $\alpha_1, \ldots, \alpha_p$ when interacting with α_k. It is α_k that reveals in some sense the possible futures of its opponents. The encoding of α_k also contains a selector S_p^k, able to extract from an opponent its own possible future. The interaction with a net $[c']$ representing a cell labeled α_i will make the selector of $[c]$ choose the "correct" package in the menu of $[c']$ and vice-versa. By a clever wiring, one obtains the translation of the corresponding right-hand side R_{α_k,α_i}. The big picture is shown in Figure 1.2.

Lafont notes that the encoding enjoys a really strong property. It is *compatible with reduction*, in other words $[\alpha_i] \bowtie [\alpha_j]$ reduces to the translation of $\alpha_i \triangleleft \triangleright \alpha_j$. Since interaction nets are deterministic up-to-interaction permutation, if a net $\mathcal{N} \to^* \mathcal{M}$, then $[\mathcal{N}] \to^* [\mathcal{M}]$.

It is however not that simple in the general case since, as you might have noticed, the translation of a cell contains translations of other cells, hidden inside the menu for all possible reductions. This means that the encoding above only suffices for *recursion-free* interaction net system in the following sense. Assume the alphabet $\alpha_1, \ldots, \alpha_p$ can be ordered in a way that any $R_{\alpha_i}^{\alpha_j}$ contains only cell with labels strictly smaller than α_i and α_j. The menu $[R_{\alpha_1}^{\alpha_k}] \& \ldots \& [R_{\alpha_p}^{\alpha_k}]$ contains the possible futures of the opponents of α_k; these are finite. But imagine one of the $R_{\alpha_k}^{\alpha_i}$ contains α_i itself?

It is anyhow possible to replace the actual infinite tree of possible futures of a cell by a potential one, and this is the real achievement of Lafont's work.

This is made possible by the nice property of packages, *i.e.* there ability to be erased and most of all duplicated. With the menu and selector system, it is possible

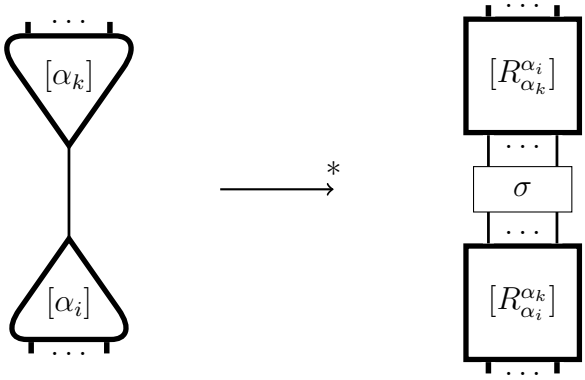


Figure 1.2: The translation is compatible with reduction

to create what Lafont calls a *genetic code of the system*. Let $\mathcal{M}_{\alpha_k}$ denote the menu $R^{\alpha_k}_{\alpha_1} \& \ldots \& R^{\alpha_k}_{\alpha_p}$.

Then we create a menu $\Gamma = \mathcal{M}_{\alpha_1} \& \ldots \& \mathcal{M}_{\alpha_p}$ and a version of it that is duplicable, denoted $!\Gamma$. We will call $\mathcal{M}_{\alpha_i}$ a *submenu* of $!\Gamma$.

Now the translation of a cell does not contain any more the translation of $\mathcal{M}_{\alpha_k}$ directly, but rather a piece of net, in the simple form of a selector, that can extract it from $!\Gamma$. It is also connected to a copy of $!\Gamma$. The translation of a cell can now make a copy of that $!\Gamma$ for personal use and extract from it the submenu corresponding to its own future. When the interaction needs to be encoded, the concerned cells extract from their respective submenus the correct half-of-rule and link each element of that to their original copy of $!\Gamma$.

In fact, the translation of a cell is just a pointer on the part of $!\Gamma$ it needs in order to complete the interaction and a decoder which first duplicates $!\Gamma$ and extracts the part it needs. Each such piece of "extracted code" contains, in turn, the copier and a set of pointers to other pieces of $!\Gamma$. This copier gives each pointer a copy of the full package.

Why does it ever stop then? Assume the recursion-free alphabet. Then, the minimal labels have no rules to them or become wirings, so they do not need nor pointers to any part of $!\Gamma$, nor a decoder. In such a case, the abusive notation $[R]$ can be any normal net. In an alphabet with recursion, the trick of the $!\Gamma$-package is that possible futures are extracted on-demand from it at each step of the simulation, so each step is indeed finitely simulated.

Let us now define Lafont's universal system for interaction nets. We leave the details for Chapter 4 where we give a universal system for multiport interaction nets which is (almost) an extension of Lafont's encoding.

Interaction combinators

To do all of the above, all Lafont needs are three cells and six simple rules that we give here to give the taste of the simplicity of the system.

- a symbol γ, called *constructor*, with arity 2:
- a symbol δ, called *duplicator*, with also arity 2:
- a symbol ε, called *erasor*, with arity 0:

The six rules are rather simple. Lafont separates them in two groups, *commutation* when the two cells are of different label — $\gamma \bowtie \delta, \gamma \bowtie \varepsilon, \delta \bowtie \varepsilon$ — and *annihilation* when they carry a same label — $\gamma \bowtie \gamma, \delta \bowtie \delta, \varepsilon \bowtie \varepsilon$. They are shown in Figure 1.3.

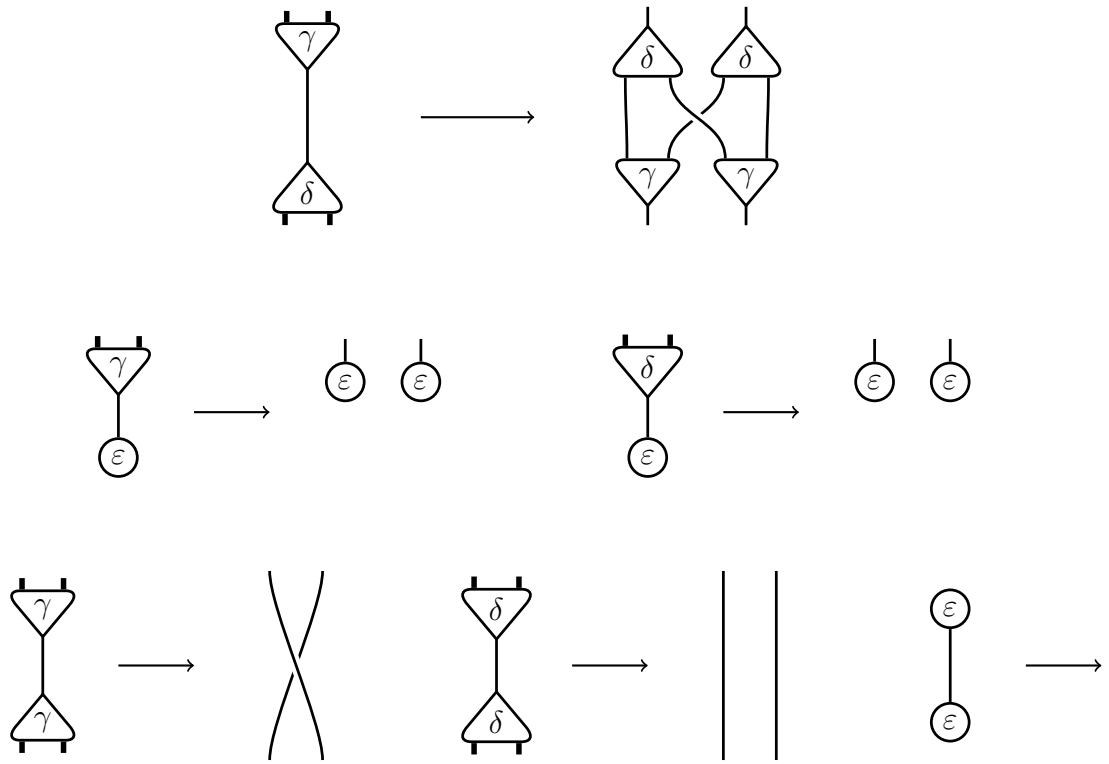


Figure 1.3: The rules of Lafont Interaction Combinators

In [24], Simon Gay gives a universal system, similar to the one of Lafont but with eight combinators. He declares himself that Lafont's solution "goes further" than his work, not only because it has less combinators, but because it deals naturally with interactions between cells of the same kind, it identifies labels which have the same rules, which seems natural, and he gets rid of the contraction node in a very elegant manner, further described in Chapter 4. On the other hand, Bechet gives in

[4] a universal system based on the one of Lafont with only two combinators and three rules, but the main rule with computational meaning is really complex: it encompasses the three $\gamma \bowtie \delta, \gamma \bowtie \gamma, \delta \bowtie \delta$ rules in one. In fact, Bechet result is more interesting in the fact that it shows how to encompass several cells in one, realizing some kind of `case`-net on which study is based Gay's system.

1.2 Non-deterministic extensions

We now define some non-deterministic extensions of Lafont's interaction nets. In the following, we will refer to the nets defined in the previous section as *Lafont nets* and *Lafont net systems*, as opposed to the extensions discussed from now on. We will refer to cell with a unique principal port as *simple cells* if this fact needs to be stressed and wires connecting at most two ports as *simple wires*. The reader can already guess how interaction nets can be extended.

Non-determinism will come from the fact that a net can reduce in at least two ways that are not brought together anymore, in other words when confluence is lost. Our purpose is to define non-deterministic systems but stay in the spirit of interaction nets. The minimal requirements for that, we believe, is the existence of primitives – *cells* with *ports* and *wires* – and a *binary, local* interaction: redexes are composed of two primitives of style *cell* connected by a wire, without any kind of priority on the rules: at each moment, the decision of which redex is reduced is completely arbitrary (strategies can be studied *a posteriori*, but not imposed to the system). Unlike Alexiev [1], we think that a port is a connection point between two elements at most. For the rest, let your imagination flow... Three major non-deterministic interaction net families emerge in the literature. Cited already, Alexiev [1], but others authors – Banach [3], Khalil and Fernandez [20], Mazza [38], Beffara and Maurel [5], Ehrhart and Regnier [17], and in some sense Honda [29] – independently introduced several non-deterministic extensions, for proof-theoretic purpose or for discussing a graphical framework for mobile and concurrent systems.

𝔄nd thus, let the story of 𝔑on-𝔇eterministic 𝔍nteraction 𝔑ets begin.

1.2.1 Multirule nets

A first and most natural extension of Lafont net systems are non-deterministic rules. In fact, one of the requests for admissible rules was that they are defined without ambiguity. It is sufficient to remove this condition to get systems which are not confluent any more. We call such systems *multirule* interaction net systems.

It is clear that any reasonable equivalence will not identify a net of a multirule system and a net of a Lafont interaction net system: the latter, whenever the computation is terminating can have only one result.

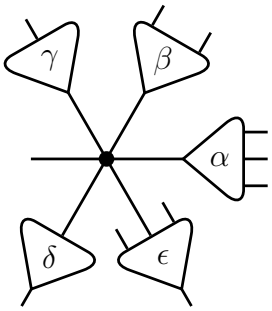
An active pair $\alpha \bowtie \beta$ has now several ways to reduce. We call *degree of non-determinism* the maximum number of ways an active pair can reduce. We usually only consider systems with finite degree of non-determinism, but the encoding result into multiwire systems will show this requirement to be rather insignificant.

We consider the possible reductions of an active pair to be numbered and denote by $\alpha \lhd_k \rhd \beta$ the RHS of the k-th rule for the active pair $\alpha \bowtie \beta$. Incompatibility of interactions is trivial: two interactions are incompatible iff they are triggered by the same active pair. Since non-determinism is defined here on interactions, it is a property of the rules alone, so the graphical representation of multirule interaction nets are the same as the one of Lafont nets.

A particular case of non-deterministic rule is the case of asymmetric rule , *i.e.* where $\alpha \lhd \rhd \alpha \neq \overline{\alpha \lhd \rhd \alpha}$. A first interesting result about multirule systems can be found in [1](p46). It states that a multirule interaction net system can be encoded into one where the only non-deterministic rule is in fact a non-symmetric reflexive one. It is enough to create a net that can "pick a random number" to choose which rule to use.

1.2.2 Multiwire nets

Another natural non-deterministic extension of Lafont nets is to allow for wires to connect more than two ports. Interactions are still due to pairs of cells connected by their principal ports. In this framework, cells are represented as in Lafont nets, but wires are now star-like edges, called *multiwires*:



This multiwire has a free port. It is a usual technicality to consider a wire to only be connected to at most one free port, but this condition can be relaxed without much trouble. A wire which is connected to a free port is said itself to be *free*. It is *bound* otherwise.

Rules now have to be slightly changed. In fact, when two cells interact, they first disconnect from the wire and then interact. This corresponds to the fact that in an interaction, the two involved principal ports disappear. Because of the requirement that the left and right hand-sides of a rule have the same interface, rules are represented with the reference to a multiwire, as represented in Figure 1.4 below.

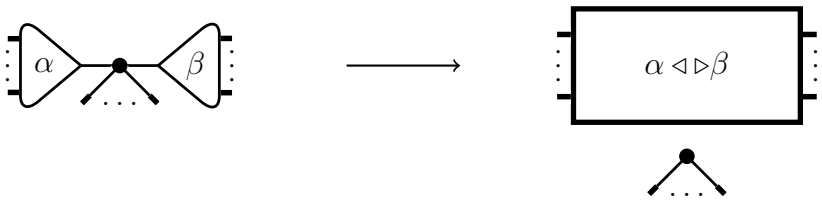
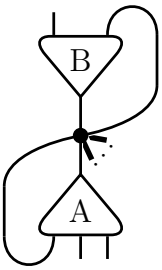
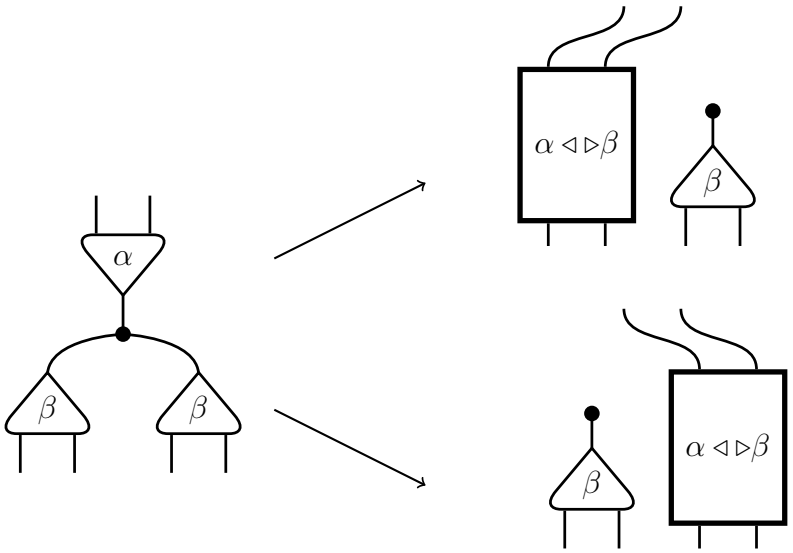


Figure 1.4: A rule in a multiwire interaction net system.

The disconnection from the wire on which is made the interaction is not a strong requirement. In the following configuration, the resulting net stays connected to what stays of the wire.



When a wire connects several active pairs, the choice of which pair interacts is non-deterministic. Imagine a rule $\alpha \bowtie \beta \to \alpha \triangleleft \triangleright \beta$, both cells having arity 2 and a net $\mathcal{N}$ composed of one α-cell and two β-cells:

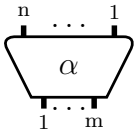


This shows incompatibility in multiwire interaction nets: two interactions are *incompatible* iff the active pairs triggering them share exactly one cell.

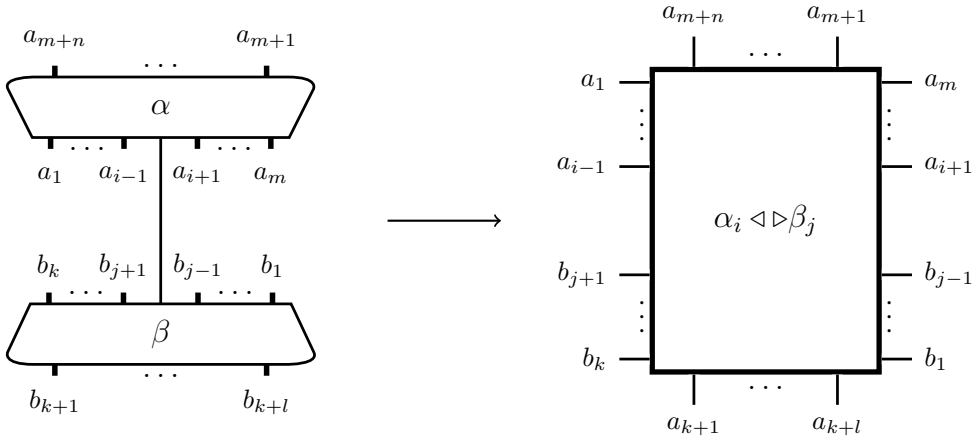
1.2.3 Multiport nets

Another natural extension of Lafont nets is given by a language of *multicells*, which not

only have several auxiliary ports, but can also have more than one principal port. Cells are now represented by trapezia. Principal ports are represented on the short parallel sides and auxiliary ones on the long parallel side. Again, ports are considered ordered clockwise on each side.

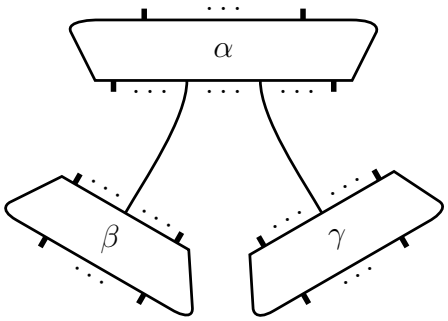


Following the definition of rules which stipulates that left-hand sides and right-hand sides have same interface, rules are represented as follows:



where $\alpha_i \lhd\rhd \beta_j$ is a multiport net. In some sense, this corresponds exactly to the generalization of a simple rule, in the following sense. One can imagine the following decomposition of the interaction above. First, the interaction transforms α (and β similarly) into a simple cell α' with, as principal port a_i and auxiliary ports $a_1, \ldots, a_{i-1}, a_{i+1}, \ldots, a_{m+n}$. Then, the two cells α', β' interact into $\alpha_i \lhd\rhd \beta_j$. Of course, this decomposition is not correct w.r.t. interaction nets, since the first interaction has a redex in its RHS.

If a multicell c is connected by its i-th principal port to the p-th principal port of a cell d and by its j-th principal port to the q-th principal port of a cell d', the choice of which interaction is triggered is non-deterministic (if they have rules of course). Both c, d and c, d' form active pairs denoted respectively $c_i \bowtie d_p$ and $c_j \bowtie d'_q$. Note that d and d' can be the same cell as long as $p \neq q$.



Here again, incompatibility comes from sharing of cells in active pairs. Two interactions are *incompatible* iff the active pairs triggering them share one or two cells.

1.3 Relative expressivity

Even though several results appear in the literature about relative expressivity of some particular interaction net systems, only two works [1, 40] address this question directly on interaction nets in general: given a system in one extension, is it always possible to find a system in this other extension that has the same expressivity? The work of Alexiev is actually an attempt to elaborate a complete study of the subject. He studies the possibilities to encode concurrent extensions in one another. Mazza, on the other hand, uses interaction nets as a case study to elaborate a general theory of relative expressivity based on event structures.

In this section, we will expose these two works, so we can later on compare our results with theirs. In fact, Chapter 3 will be similar to Alexiev's thesis. The second part of this section exposes event structures (very briefly) and Mazza's results of separation that Alexiev did not obtain. In a third part, we give a new separation result based on Mazza's technique, and that is a separation between multiport and multiwire nets.

1.3.1 Encodings among non-deterministic interaction nets

In his PhD thesis, Alexiev [1] studies the possibility to express one extension of interaction nets into another. Even though the names he gives to the systems are different, he discusses the three extensions we have given above, and also a fourth one, in which ports can be shared by more than two agents (more precisely by an agent and several simple wires). It seems at first to be the exact same thing as the multiwire setting, but it is in fact closer to the multiport setting. So close that we do not look into that extension in this work. The other important reason to discard this framework is that it breaks the soul purpose of ports as we wish to study them: as primitives for names (see Chapter 3).

Alexiev's work is rather exhaustive on the extensions he defines: he gives translations and separations for all pairs of them, even if not explicitly for each. Some translations are just hinted about (not the most straightforward ones as we will see), and the whole machinery is not very formal. The main issue is that his criteria for translation are rather vague and at the same time quite strict. Following Lafont, he considers translations have to be *complete* in a classical way: every single-step from $\mathcal{M}$ to $\mathcal{N}$ is emulated by a proper multistep from the translation of $\mathcal{M}$ to the translation of $\mathcal{N}$. This already is a strong requirement, which is awesome for encodability results, but that we cannot accept for separation results. Even more arguable is his definition of *soundness*. Because he does not use any notion of semantical equivalence such as bisimulation, he has to use a machinery of *completion context* to deal with what he calls *pre-commitment steps* and we would call *garbage steps* (because they only pose problem at the end of computation): some steps in the target system which have no

counterpart in the original one and can occur after the original net has finished its computation. Such steps are implicitly dealt with by semantical equivalences based on bisimulations. Finally, his criteria for *uniformity* and *preservation of port type* are too strong for a separation result as often discussed in the literature (*e.g.* [26]). Luckily, most of this results are positive, so the strong criteria are a good justification of the translations. Unluckily, the lack of a proper semantical equivalence yields translations which have poor properties. Most of them introduce divergence and do not even verify the criteria given by the author himself, such as preservation of port type.

We will of course not detail the translations here, but reproduce the summary result of his thesis. Not to get confused, we will define the terminology for systems which differs slightly from his:

- InMR for Multirule: an active pair can have several rules applied randomly;
- InMP for Multiport: cells can have several principal ports;
- InHP for Hyperport: ports can be connected to several (simple) wires;
- InMW for Multiwire: wires can connect any number of ports;

Note that Alexiev does not consider combinations of systems, so for instance MR, MP and HP have simple wires, etc.

Table 1.1: Inter-representation of Non-Determinism: Basic Ideas

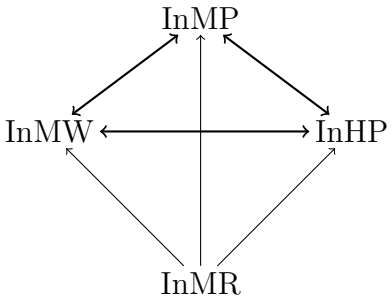
Source	Target	Status	Description and critics
InMR	InMP	Given	Uses *self-commitment*: each cell decides which rule to use (incompatibility is easily taken care of without returning to initial state). Elegant, but self-commitment can happen even if cell is not part of a cut: not strongly sound. Can be easily improved to be so.
InMR	InHP	Given	Same translation as above where multiple principal ports are merged into a unique hyper port. Same defects and can be corrected in the same way.
InMR	InMW	Hinted	Same exact remark as before. Moreover, uses rules where multiwire on which is made the cut stays connected to the RHS. Can be improved to avoid that also.

Source	Target	Status	Description and critics
InMP	InMR	Does not exist	The three separation results for translations **into** InMR are based on the same argument: in InMR a free principal port stays such for ever, while in the other systems it is not the case. We believe this gives to much importance to the type of the the port, principal or auxiliary.
InMP	InHP	Given	Uses *commitment* of cells to a particular active pair. Is only developed the case where commitments are compatible. Incompatibility is not taken care of and the translation doesn't seem to be extended to do so. It would anyway introduce divergence.[1]
InMP	InMW	Given	Uses *partial* and *full commitments*. Every principal port is transformed into a cell that listens to the outside for possible interaction and one that listens for the inside for a possible *passivation*: transform a principal port into an auxiliary one. Introduces divergence[1] when commitments do not match (a general problem of agreement translations).
InHP	InMR	Doesn't exist	See InMP → InMR.
InHP	InMP	Hinted	Transforms each hyperport of a cell A of degree n into a cell a attached to the representative of A and to n cells each representing a connection. Cell a can transmit messages from A to port cells and backwards. (The hint wasn't enough for me to understand).
InHP	InMW	Given	Basically the same as InMP to InMW. Shows the similitude between InMP and InHP. Introduces divergence.[1]
InMW	InMR	Doesn't exist	See InMP → InMR.

[1] *Introducing divergence* means in this case that there exists an interaction net system in the source language for which there is a net that has only finite reduction paths but any translation into a system of the target language yields a net with an infinite reduction path.

Source	Target	Status	Description and critics
InMW	InMP	Hinted	Represents a multiwire by a communication zone. Correct, but as we will see in Chapter 3, it is not as straightforward as it seems.
InMW	InHP	Given	Replace a multiwire by a *connector* cell which chooses two compatible cells connected to it and creates an active pair, which then simulates the original interaction. To deal with the translation of merging two multiwires into one, the author extends the system with a multirule, which he thinks is unavoidable.[2]

Which can be in its turn summarized as follows, where $\mathcal{S} \to \mathcal{T}$ means there exists an encoding of $\mathcal{S}$-nets in $\mathcal{T}$-nets:



If one does not take care about details, one could consider InMP, InMW and InHP to be equipotent in the sense of encodability. The comments in Table 1.1 show that some translations are arguable, and most of all, almost none are justified by any semantical equivalence. We actually believe any reasonable equivalence would justify them, except for divergence sensitive equivalences. It is often discussed if an encoding that introduces divergence is acceptable or not ([44] vs. [45] for instance). Finally, the encoding InMW into InHP is not strictly valid since it uses a multirule. We know how to correct that easily, so we consider the translation to be valid.

Taking into account divergence, we can give an order on the systems where $A < B$ stands for: any system of framework A can be encoded into a system of framework B but not backwards, and $A \leq B$ stands for: any system of framework A can be

[2] Strange since the author has knowledge of the InMW to InMP translation that can be easily adapted here (without any need for explicit connectors) and where communication zones deal with the merging of multiwires.

encoded into a system of framework B and the translation from B to A given by Alexiev introduces divergence:

$$\text{InMR} \quad < \quad \text{InMW} \quad \leq \quad \text{InHP} \quad \leq \quad \text{InMP}$$

And in fact, this is what we will show more formally in Chapter 3 (except for InHP). We will also study the different combinations of systems, as systems with multirules and multiport cells, or multiport cells and multiwires.

1.3.2 A separation technique based on Event Structures

We now introduce a technique proposed by Mazza [40] as an abstract way of comparing rewriting systems. It is based on Winskel's *event structures* [58] that represent the dynamics of rewriting by considering each rewriting step as an event. Events are then related by causality and incompatibility. It is the complexity of such graphs of events that give is some sense a measure of the complexity of the dynamics of the system. Mazza adds to this construction a notion of *embedding*, describing in this way that an event structure contains another one. For a system to be able to generate structures containing the structures of another system is considered as a criterion for superiority in the expressivity hiererachy. Inversely, its impossibility is considered to separate systems in that hierarchy.

Without entering into details, let us now see some basic definitions and some basic results as a pre-taste of the separation result we give in Section 1.3.3. The end of this section is almost all taken directly from the cited article by Mazza. In what follows, if $(X, \leq)$ is a poset and $u \subseteq X$, we denote by $\downarrow u$ the downward closure of u, *i.e.* $\downarrow u = \{y \in X \mid \exists x \in u, y \leq x\}$ and we write $\downarrow x$ for $\downarrow\{x\}$. We also denote by $\mathcal{P}_{\text{fin}}(X)$ finite subsets of X.

Definition 1.3.1 (Event Structures (Winskel and Nielsen, 1995))**.** An *event structure* is a triple $E = (|E|, \leq, \smile)$ where

- $|E|$ is a set, the elements of which are called *events* and are ranged over by $a, b, c, \ldots$
- $\leq$ is a partial order on $|E|$, called *causal order*, s.t. for all $a \in |E|$, $\{x \in |E| \mid x \leq a\}$ is finite
- $\smile$ is an anti-reflexive and symmetric relation on $|E|$, called *conflict relation*, such that for all $a, b, c \in |E|, a \smile b \leq c$ implies $a \smile c$.

The condition on the relation $\smile$ is exactly the one that forbids duplication of events.

We say that u is a *configuration* of E iff $\downarrow u = u$ and $a, b \in u$ implies $a \not\smile b$, denoted $a \frown b$. The set of finite configurations of E is denoted by $\mathcal{C}(E)$ and ranged over by $u, v, w, \ldots$ If u is a configuration and a an event such that $a \notin u$ and $u' = u \cup \{a\}$ is

a configuration, we say u *enables* a. The smallest configuration enabling $a \in |E|$ is $\downarrow a \setminus \{a\}$ denoted $\lceil a \rceil$.

Let $E = (|E|, \leq, \smile)$ and $E' = (|E'|, \leq', \smile')$ be event structures and $R \subseteq |E| \times |E'|$. We denote by $\pi_1(R), \pi_2(R)$ the projections of R. If $u \in \mathcal{C}(E)$, $a \in |E|$ is enabled by u and $v = u \cup \{a\}$, we write $u \xrightarrow{a}_R v$ if $a \in \pi_1(R)$ (we call this a *computational transition* labeled by a) and $u \rightarrow_R v$ otherwise (we call this an *administrative transition*). They correspond in some sense to visible and invisible transitions. We denote by $\Rightarrow_R$ the reflexive transitive closure of $\rightarrow_R$ and we write $u \stackrel{a}{\Rightarrow}_R v$ if there exist $u', v' \in \mathcal{C}(E)$ s.t. $u \Rightarrow_R u' \xrightarrow{a}_R v' \Rightarrow_R v$. We apply the same notation for E'. Moreover, given $u \in \mathcal{C}(E)$ and $u' \in \mathcal{C}(E')$, we set $supp_R(u) = u \cap \pi_1(R)$ and $supp_R(u') = u' \cap \pi_2(R)$.

Definition 1.3.2 (R-bisimulation)**.** Let $E = (|E|, \leq, \smile)$ and $E' = (|E'|, \leq', \smile')$ be event structures and $R \subseteq |E| \times |E'|$. An *R-bisimulation* between E and E' is a relation $\mathcal{B} \subseteq \mathcal{C}(E) \times \mathcal{P}_{\text{fin}}(R) \times \mathcal{C}(E')$ such that $(\emptyset, \emptyset, \emptyset) \in \mathcal{B}$ and, whenever $(u, \phi, u') \in \mathcal{B}$, we have:

1. ϕ is a poset isomorphism between $(supp_R(u), \leq)$ and $(supp_R(u')$;
2. $u \xrightarrow{a}_R v$ implies $u' \stackrel{a'}{\Rightarrow}_R v'$ with $(v, \phi \cup \{a, a'\}, v') \in \mathcal{B}$;
3. $u \rightarrow_R v$ implies $u' \Rightarrow_R v'$ with $(v, \phi, v') \in \mathcal{B}$;
4. $u' \xrightarrow{a'}_R v'$ implies $u \stackrel{a}{\Rightarrow}_R v$ with $(v, \phi \cup \{a, a'\}, v') \in \mathcal{B}$;
5. $u' \rightarrow_R v'$ implies $u \Rightarrow_R v$ with $(v, \phi, v') \in \mathcal{B}$;

We say that E and E' are *R-bisimilar*, and we write $E \approx_R E'$, if there exists an R-bisimulation between them.

The meaningfulness of an R-bisimulation depends on R: for example, for any E, E', $\{(u, \emptyset, v)\}$ is a $\emptyset$-bisimulation. To avoid this kind of degeneracy, we consider special cases of R-bisimulation, where R is in some sense maximal.

Definition 1.3.3 (Bisimilar embedding[3])**.** Let $E = (|E|, \leq, \smile)$ and $E' = (|E'|, \leq', \smile')$ be event structures. A *bisimilar embedding* of E into E' is a relation $\iota \subseteq |E| \times |E'|$ such that:

totality: $\pi_1(\iota) = |E|$;

injectivity: for all $a, b \in |E|$, $\iota(a) \cup \iota(b) \neq \emptyset$ implies $a = b$;

bisimilarity: $E \approx_\iota E'$.

We write $E \stackrel{\iota}{\hookrightarrow} E'$ to denote the fact that ι is an embedding of E into E' or simply $E \hookrightarrow E'$ to state the existence of an embedding.

[3]The reader who is familiar with history-preserving bisimulations will recognize that E can be embedded into E' precisely when, once we consider the events to be labeled by themselves, there is a way of labeling the events of E' over $E \cup \{\tau\}$ so that E and E' are weakly history-preserving bisimilar.

Notice that embeddings, even though based on bisimulation, are *not* symmetric. This is a desirable feature since their purpose it to give some order on expressivity: some systems are strictly more powerful than others.

We now define some particular substructures that can be "found" in event structures that allow to give some separation results.

Definition 1.3.4 (Immediate conflict)**.** Let E be an event structure. We say that $a, a' \in |E|$ are in *immediate conflict*, and we write $a \sharp a'$, iff $a \smile a'$ and there exists a configuration enabling both a and a'. We denote by $\overset{\sharp}{\sim}$ the reflexive transitive closure of $\sharp$.

Definition 1.3.5 (Confusion)**.** Let E be an event structure. A *confusion of type I* in E is a triple $(a, b, c) \in |E|^3$ such that $a \neq c$, $a \sharp b$, $b \sharp c$ and a and c are *not* in immediate conflict. A *confusion of type II* in E is a pair $(a, b) \in |E|$ such that $a \sharp b$ and $\lceil a \rceil \neq \lceil a \rceil$.

Proposition 1.3.6. *Let E, E' be two event structures with E containing a confusion. Then E is embeddable in E' implies that E' contains a confusion.*

It is possible to assign to any interaction net $\mathcal{M}$ of any INS an event structures $Ev(\mathcal{M})$ that describes the reductions of $\mathcal{M}$. One can then notice that if $\mathcal{M}$ is a net of a multirule system $\mathcal{S}$, then $Ev(\mathcal{M})$ is confusion-free; in fact, in multirule nets all incompatible interactions are triggered by a same active pair. On the other hand, all concurrent calculi exhibit confusion in the event structures corresponding to some of there terms. So do multiwire and multiport nets; an example is given in Figure 1.5. The last proposition then says that no bisimilar embedding can be found from any such net to any net of any multirule system, suggesting that these later are less expressive.

with an only rule for $\alpha \bowtie \beta$ with a rule for $\alpha_1 \bowtie \alpha_2$

Figure 1.5: Example of nets which induce confusion of type I in their corresponding event structure.

One would expect that if $\mathcal{M}, \mathcal{M}'$ are two computational models for which there is an encoding from $\mathcal{M}$ to $\mathcal{M}'$ and for which one can build event structures, that it would be the case that $\mathcal{M} \hookrightarrow \mathcal{M}'$. It is the case for instance for Lafont interaction nets into the Interaction Combinators [33]. But it isn't true of Mazza's encoding of π-calculus in multiport interaction nets: it does not induce a bisimilar embedding [38]. Ehrhard and Laurent give an encoding of π-calculus into differential interaction

nets [16], but π-calculus admits confusion while differential interaction nets do not. By Proposition 1.3.6, it does not induce a bisimilar encoding. Worse, no encoding of π-calculus into differential nets do. Conversely, does the existence of a bisimilar embedding between the event structures of two computational models say anything on the existence of encodings? Probably not.

Mazza's results give some insight on the profound reasons of some separation properties. The generality of the framework is very pleasant and promises some nice further developments.

1.3.3 Event Structures of multiport vs. multiwire systems

Even though one can question the usability of bisimilar embeddings, we wish to give another separation result based on that technique, as it hints on a separation result of Chapter 3. We shall compare multiwire and multiport interaction net systems.

A bisimilar embedding is said to *introduce divergence* if, whenever $\mathcal{B}$ is a bisimulation associated with ι, there exists $(u, \phi, u') \in \mathcal{B}$ such that there is an infinite sequence of administrative transitions $u' \to_\iota u'_1 \to_\iota u'_2 \to_\iota \dots$ in E'. We show that some configurations are preserved by bisimilar embeddings that do not introduce divergence.

But first, we define for any net a graph of conflicts between the possible interactions in it. Formally, we find the conflict-graph $\mathtt{Cg}(\mathcal{N})$ of a net $\mathcal{N}$ in the following way. We name each cell of $\mathcal{N}$. A vertex of $\mathtt{Cg}(\mathcal{N})$ is a triplet (a, b, k) where a, b are the names of the cells of the active pair and k the number of the rule for this interaction. We then add an edge between any two vertices that share a name in their label.

Conflict graphs are related to event structures because their nodes are the "events" which are possible in $\mathcal{N}$ and the edges of $\mathtt{Cg}(\mathcal{N})$ represent conflict between events. Causality is left appart as all the considered events are possible at the same time. An example is given in Figure 1.6, for a multiport interaction net system with simple rules and a net in which all cuts are active pairs.

In event structures, configurations correspond exactly to nets, thus the following definition.

Definition 1.3.7 (Conflict graph)**.** Let $E = (|E|, \leq, \smile)$ be an event structure and $u \in \mathcal{C}(E)$ a configuration of E. The *conflict graph of* u, noted $\mathtt{Cg}(u)$ is the graph which vertices are events triggered by u and edges represent the conflict relation.

We can easily define the conflict graphs that can be generated by some interaction net systems. For instance, a conflict graph of any Lafont net is a discrete graph. In fact, Lafont nets are conflict-free. More interesting are conflict graphs of multirule interaction nets. These are graphs composed of disjoint cliques.[4] The situation is more complex for multiwire and multiport graphs. To each cell name corresponds a clique. Moreover, since interactions are binary, a vertex can belong to at most two such cliques.

[4] Cliques in conflict graphs correspond to anticliques in event structures.

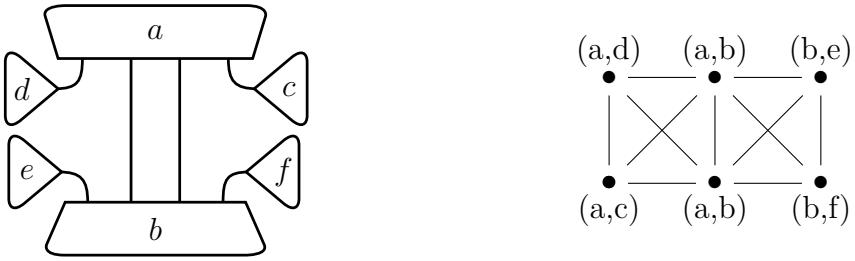


Figure 1.6: A multiport net and its conflict graph (all cuts are active pairs).

Definition 1.3.8 (Entangled event structure)**.** Let G be a graph. A *clique covering* of G is a family $\mathcal{C}$ of cliques of G such that every edge of G belongs to at least one clique in $\mathcal{C}$. A *good covering* of G is a clique covering for which each vertex v of G belongs to at most 2 cliques. G is *entangled* if, for every good covering of G, there exist cliques C, C' such that $|C \cap C'| > 1$.

An event structure E is *entangled* if there exists a configuration $u \in \mathcal{C}(E)$ such that $\mathtt{Cg}(u)$ is entangled.

The conflict graph of a multiwire or multiport net has (at least one) good covering, corresponding to the clique decomposition given above[5]. On the other hand, conflict graphs of multiwire nets cannot be entangled. In fact, in multiwire systems, each pair of cells can lead to at most one interaction so two vertices of its conflict graph cannot share both their names.

Lemma 1.3.9. *Let $E \overset{\iota}{\hookrightarrow} E'$ and let $d, e \in |E|$, $d' \in \iota(d)$ and $e' \in \iota(e)$. Then $d \smile e$ iff $d' \smile' e'$.*

Proof. It is an immediate consequence of the properties of ι-bisimulations. If d and e are not in conflict, then by bisimulation d' and e' are not either and vice-versa. □

We say G' is *a full subgraph* of G if, for any edge e of G with extremities $a, b \in G'$, e is an edge of G'.

Lemma 1.3.10. *Let G be a graph and H a full subgraph of G. Then H is entangled implies that G is entangled.*

Proof. Let $C_1, \ldots, C_n$ be a good covering of G. Since H is a full subgraphs, $C_1 \cap H, \ldots, C_n \cap H$ is a clique covering of H. By hypothesis, there exist $i, j \leq n$ such that $|(C_i \cap H) \cap (C_j \cap H)| > 1$. But $1 < |(C_i \cap H) \cap (C_j \cap H)| = |C_i \cap C_j \cap H| \leq |C_i \cap C_j|$ so G is entangled. □

[5]If we associate to an interaction net N a multigraph G in which vertices are cells of N and edges interactions triggered by pairs of cells, the conflict graph of N is nothing else that the line-graph of G. In [31], the authors show that a graph has a good covering if and only if it is the line graph of a multigraph. Moreover, they show that the existence of a good covering is a polynomially solvable problem.

Proposition 1.3.11. *Let $E \overset{\iota}{\hookrightarrow} E'$ without introducing divergence, and E entangled. Then E' is entangled.*

Proof. Let $\mathcal{B}$ be a bisimulation associated with ι and let u be a configuration of E such that $\mathtt{Cg}(u)$ is entangled. Let $a_1, \ldots, a_n$ be the vertices of $\mathtt{Cg}(u)$. There must be a configuration $u' \in \mathcal{C}(E')$ such that $(u, \varphi, u') \in \mathcal{B}$ for some isomorphism φ, and since ι does not introduce divergence, there exists a maximal sequence of administrative transitions $u \Rightarrow_\iota u''$ such that there is no administrative transition starting from u''. Then, since $\mathcal{B}$ is a bisimulation, and since for all $1 \leq i \leq n$, we have $u \xrightarrow{a_i}_\iota u \cup \{a_i\}$, we must have, for all $1 \leq i \leq n$, some $a'_i \in \iota(a_i)$ such that $u'' \xrightarrow{a'_i}_\iota u'' \cup \{a'_i\}$. By Lemma 1.3.9, $\mathtt{Cg}(u'')$ contains a copy of $\mathtt{Cg}(u)$ as full subgraph, and by Lemma 1.3.10 $\mathtt{Cg}(u'')$ is entangled. □

This means that divergence free event structures corresponding to multiport interaction nets cannot all be embedded in divergence free event structures of multiwire interaction nets. The separation is relevant: the conflict graph of the multiport net given in Figure 1.6 is entangled. A system that provides such a net is one containing cells α with 4 principal ports and β with 1, and rules for $\alpha_2 \bowtie \alpha_3$, $\alpha_1 \bowtie \beta_1$ and $\alpha_4 \bowtie \beta_1$.

Chapter 2

Structural Operation Semantics for Interaction Nets

In the present chapter, we try to provide interaction nets with a proper operational semantics. In particular we describe a CCS-*like* labeled transition semantics for graph transformation systems.

We first discuss a straightforward out-of-the-blue definition of a SOS-semantic for simple interaction nets. The problems we meet make us try another approach, that of hypergraph rewriting, considering that interaction nets can be seen as particular hypergraphs. For this, we recall the basic definitions and concepts for the concrete case of hypergraphs in Section 2.2; in particular we give a brief review of the Borrowed Context technique. In Section 2.3, we provide a reformulation of the Borrowed Context technique in analogy to Milner's CCS; however, this analogy is imperfect as there is no need for a counterpart of the communication rule. This issue is addressed in Section 2.4, where we present our main results, which allow to define a graph transformation counterpart of a communication rule. These results are applied in Section 2.5 to obtain a satisfactory SOS like reformulation of the Borrowed Context technique. We conclude in Section 2.6 with some restrictions on hypergraphs and their rules that allow for a simplified, and sometimes usable definition of the communication rule derived in previous sections in the framework of interaction nets. This definition allows us to formalize the initial intuitive SOS-semantics.

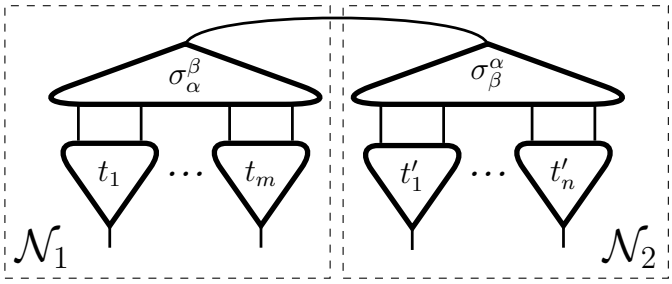
The concepts for this chapter are introduced in the paragraph about operational semantics (p. 10).

2.1 SOS for simple interaction nets

We try to give an SOS to simple interaction nets. We restrict to the simple case, because of the property expressed in Proposition 1.1.5 that we remind below: it is possible to define the half of an interaction that comes from each cell of the rule.

We then consider that an action corresponds to rewriting a cell with free principal port into its half, considering each rule it is involved with. This is reminiscent of the CCS-approach, present in the π-calculus and its derivatives.

Proposition 2.1.1 (Splitting)**.** *Let $\mathcal{S}$ be a simple interaction net system containing the cells γ, δ of Lafont interaction combinators, $\alpha \bowtie \beta$ be an active pair of $\mathcal{S}$ and $\alpha \triangleleft \triangleright \beta$ the right-hand side of the rule for this pair. Let $a_1, \ldots, a_m$ be the ports of α not involved in the interaction and $b_1, \ldots, b_n$ the ones of β. Then their is unique net $\mathcal{N}$ of the following form*



such that $\mathcal{N} \to^* \alpha \triangleleft \triangleright \beta$, where the roots of $t_1, \ldots, t_m$ are $a_1, \ldots, a_m$ and those of $t'_1, \ldots, t'_n$ are $b_1, \ldots, b_n$.

In the following, we will refer to $\mathcal{N}_1$ as R_α^β and to $\mathcal{N}_2$ as R_β^α. The first idea is to define actions on a free port labeled by the cell that can carry out the interaction:

$$\mathcal{N} \xrightarrow{\alpha \triangleright \ on \ p} \mathcal{N}\{R_\alpha^\beta/\alpha\}$$

where p is the principal port of an α-cell in $\mathcal{N}$, the roots of the trees $t_1, \ldots, t_m$ are connected according to the port names of the α-cell and the root of σ_α^β is connected to (the residue of) p.

A first try

Now, we can write derivation rules for an operational semantics. We use graph union parametrized by a set of equalities on ports, $\mathcal{N} \cup_{\{a=a', \ldots\}} \mathcal{M}$, to denote the operation of "plugging" $\mathcal{N}$ and $\mathcal{M}$ together with at least a wire between a in $\mathcal{N}$ and a' in $\mathcal{M}$, etc. Ports not explicitly mentioned can be interconnected or not. The following rule has obviously a symmetric version:

$$\frac{\mathcal{N} \xrightarrow{\alpha \triangleright \ on \ p} \mathcal{N}'}{\mathcal{N} \cup_{\mathcal{E}} \mathcal{M} \xrightarrow{\alpha \triangleright \ on \ p} \mathcal{N}' \cup_{\mathcal{E}} \mathcal{M}} \quad \text{where } p \text{ does not appear in } \mathcal{E}$$

This rule is valid since, if not connected during the plugging operation, a free port stays free. If it is a principal port of a cell, an action can still be triggered on it.

How about a pendant of the communication rule? If their is a rule for $\alpha \bowtie \beta$ and one net has α with free principal port and the other has β with free principal port, the two cited ports can be connected together to form a bigger net which can reduce,

internally. This seems quite close to CCS-like operational semantics. It should look something like:

$$\frac{\mathcal{N} \xrightarrow{\alpha\triangleright \ on \ p} \mathcal{N}' \qquad \mathcal{M} \xrightarrow{\beta\triangleright \ on \ q} \mathcal{M}'}{\mathcal{N} \cup_{\{p=q,\ldots\}} \mathcal{M} \ \rightarrow \ \mathcal{N}' \cup_{\{p=q,\ldots\}} \mathcal{M}'}$$

This would be nice. But imagine now the system also has a rule for $\alpha \bowtie \gamma$. It could be that the action on $\mathcal{N}$ was triggered by the $\alpha \bowtie \gamma$ rule, in which case the conclusion interaction is not correct, since the plugging of $\mathcal{N}$ and $\mathcal{M}$ yields a cut on β and γ, which might not even have a rule, or at least completely different from the one for $\alpha \bowtie \beta$.

So we need the information about what piece of net was connected to trigger the interaction which justifies the action. If we denote by $\triangleleft\beta \ on \ p$ the fact of plugging a cell labeled β on port p, we should have something like

$$\frac{\mathcal{N} \xrightarrow{\triangleleft\beta \ on \ p} \mathcal{N}' \quad \mathcal{M} \xrightarrow{\triangleleft\alpha \ on \ q} \mathcal{M}'}{\mathcal{N} \cup_{\{p=q,\ldots\}} \mathcal{M} \ \rightarrow \ \mathcal{N}' \cup_{\{p=q,\ldots\}} \mathcal{M}'}$$

But now again, we do not know if in $\mathcal{M}$, q was the the principal port of a β or a γ-cell, making the conclusion interaction false in the latter case.

So it seems one needs the information about the whole rule used to justify the action. We would write something like $\mathcal{N} \xrightarrow{\alpha\bowtie\beta \ on \ p} \mathcal{N}'$ to express that when a β-cell is plugged to the port p which is the principal port of an α-cell in $\mathcal{N}$, $\mathcal{N}$ evolves into $\mathcal{N}'$ (Note that in this case, the active pair is considered ordered).

Then the communication rule could be:

$$\frac{\mathcal{N} \xrightarrow{\alpha\bowtie\beta \ on \ p} \mathcal{N}' \qquad \mathcal{M} \xrightarrow{\beta\bowtie\alpha \ on \ q} \mathcal{M}'}{\mathcal{N} \cup_{\{p=q,\ldots\}} \mathcal{M} \ \rightarrow \ \mathcal{N}' \cup_{\{p=q,\ldots\}} \mathcal{M}'}$$

Remarks

First, we need to notice that in order to obtain such a nice communication rule, we had to add in the system two new cell. One can try to do without it naively, since the splitting of the RHS of a rule is still valid, with the difference that the contact zone between the two halves can have arbitrary size. In this case, in the communication rule

$$\frac{\mathcal{N} \xrightarrow{\alpha\bowtie\beta \ on \ p} \mathcal{N}' \qquad \mathcal{M} \xrightarrow{\beta\bowtie\alpha \ on \ q} \mathcal{M}'}{\mathcal{N} \cup_{\{p=q,\ldots\}} \mathcal{M} \ \rightarrow \ \mathcal{N}' \cup_{\mathcal{E}} \mathcal{M}'}$$

the set $\mathcal{E}$ can be rather complex and can certainly not be derived from the labeled transitions, neither the ones in the premise, nor the first part of the conclusion. The resulting rule is not exactly *structural*, meaning that the conclusion can be derived straight from the premises.

The simplicity of the result can anyhow justify the extension of systems to containing also cells γ, δ and their rules. But it raises a few questions about what rules should be given for interaction between γ, δ-cells and other cells of the system? It might be that having such rules extends the system beyond reasonable limits. It might also be that forbidding γ, δ-cells to interact with any other cells is too restrictive.

Another question regards the case of multiwires. In this framework, wirings between a set of ports $\tilde{z}$ and set of ports $\tilde{z}'$ do not correspond to permutations, but rather to relations on $\tilde{z} \times \tilde{z}'$. It might not be so simple to build some tiny alphabet to derive any such relation from two predefined nets (even less, trees). It is anyhow an interesting question, that we do not adress in this work.

To cope with these problems, we will try to find help in the field of graph-rewriting, and particularly of so called *double-pushout graph rewriting with borrowed contexts*. For a detailed study of the subject, see for instance [18] by one of the key figures in the domain.

2.2 Graph rewriting : preliminaries

We first recall the standard definition of (hyper-)graphs and a formalism of transformation of hypergraphs (following the double pushout approach). We also present the labeled transition semantics for hypergraph transformation systems that has been proposed in [19]. In the present work, the more general case of categories of graph-like structures is not of central importance. We avoid category theoretical jargon and present all necessary concepts concretely for hypergraphs.

2.2.1 Hypergraphs

Definition 2.2.1 (Hypergraph)**.** Let Λ be a set of *labels* with associated *arity* function $\mathrm{ar}\colon \Lambda \to \mathbb{N}$. A *($\Lambda$-labeled hyper-) graph* is a tuple $G = (E, V, \ell, \mathrm{cnct})$ where E is a set of *(hyper-) edges*, V is a set of *vertices* or *nodes*, $\ell\colon E \to \Lambda$ is the *labeling function*, and cnct is the *connection* function, which assigns to each edge $e \in E$ a finite sequence of vertices $\mathrm{cnct}(e) = v_1 \cdots v_n$ where $\mathrm{ar}(\ell(e)) = n$, i.e. $\mathrm{cnct}(e)$ is a function from $\{i \mid 0 < i \leq \mathrm{ar}(\ell(e))\}$ to V. For a given edge $e \in E$, the set of all *e-adjacent vertices* is $\mathrm{adj}(e) = \{\mathrm{cnct}(e)(i) \mid 0 < i \leq \mathrm{ar}(\ell(e))\}$ and for a given node $v \in V$, the set of edges incident to it is $\mathrm{inc}(v) = \{e \in E \mid v \in \mathrm{adj}(e)\}$. The *degree* of a node $v \in V$, written $\deg(v)$, is the number of edges incident to it, i.e. $\deg(v) = |\mathrm{inc}(v)|$ (where for any finite set M, the number of elements of M is $|M|$). We also write $v \in G$ and $e \in G$ if $v \in V$ and $e \in E$ to avoid clutter.

Example 2.2.2 (Hypergraph). An example of an $\{\alpha, \beta, \gamma\}$-labeled hypergraph is illustrated in the grey box in Figure 2.1. The arities of α, β and γ are 2, 3 and 1, respectively. The graph has hyperedges of each "type", which are depicted as rounded

boxes with the respective label inside; moreover, the graph has four nodes. The order of the nodes that are connected to an hyperedge is usually not important (but could be fixed easily by counting counter-clockwise from the bottom left corner of the edge).

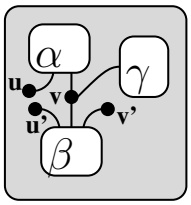


Figure 2.1: A siple hypergraph with 3 hyperedges and 4 nodes.

We usually do not discern isomorphic graphs (which roughly corresponds to the practice to consider terms of process calculi up to structural congruence); thus, we usually do not mention the "names" of nodes and edges. The full details of the above graph would be $(\{e, e', d\}, \{u, v, u', v'\}, \ell, \mathrm{cnct})$ where $\ell = \{e \mapsto \alpha, e' \mapsto \beta, d \mapsto \gamma\}$ and $\mathrm{cnct} = \{e \mapsto uv, e' \mapsto v'vu', d \mapsto v\}$.

For completeness' sake, we recall the standard definitions of hypergraph morphism, sub-graph and isomorphism. However, usually, inclusions of sub-graphs in illustrations are the obvious ones that preserve the (relative) positioning of nodes and edges.

Definition 2.2.3 (Hypergraph morphisms, inclusions, isomorphisms)**.**
Let $G_i = (E_i, V_i, \ell_i, \mathrm{cnct}_i)$ $(i \in \{1, 2\})$ be hypergraphs; a *hypergraph morphism* from G_1 to G_2, written $f \colon G_1 \to G_2$, is a pair of functions $f = (f_E \colon E_1 \to E_2, f_V \colon V_1 \to V_2)$ that preserves labels and connectivity of edges: The equality $\ell_2 \circ f_E = \ell_1$ holds and we have $f_V(\mathrm{cnct}_1(e)(i)) = \mathrm{cnct}_2(f_E(e))(i)$ for each edge $e \in E_1$ and every $i \in \mathbb{N}$ such that $0 < i \leq \mathrm{ar}(\ell(e))$.

A hypergraph morphism $f = (f_E, f_V) \colon G_1 \to G_2$ is *injective (an isomorphism)* if both f_E and f_V are injective (bijective); it is *the inclusion (of G_1 into G_2)* if both $f_E(e) = e$ and $f_V(v) = v$ hold for all $e \in E_1$ and $v \in V_1$ and then G_1 is a *sub-graph* of G_2. As usual, we write $G_1 \cong G_2$ if there is an isomorphism $f \colon G_1 \to G_2$. We write $G_1 \hookrightarrow G_2$ or $G_2 \hookleftarrow G_1$ (and very often just $G_1 \to G_2$ or $G_2 \leftarrow G_1$) if G_1 is a sub-graph of G_2.

In this subsection we have recalled the basic terminology for hypergraphs; next we shall review a standard approach to graph transformation.

2.2.2 Standard graph transformation

The most established approach to graph transformation is double pushout rewriting. It is most succinctly defined using the basic category theoretical notion of pushout, which makes it a uniform approach for arbitrary graph-like structures. However, for the particular case that we are interested in, pushouts can be understood as a variation

of the disjoint union of hypergraphs. In the next definition, we also cover the particular case of pullbacks that we shall need to properly present the Borrowed Context technique [19] in Section 2.2.3 without a purely category theoretical perspective.

Definition 2.2.4 (Pullbacks & pushouts of inclusions)**.** Let $G_i = (E_i, V_i, \ell_i, \mathrm{cnct}_i)$ $(i \in \{0,1,2,3\})$ be hypergraphs and let $G_1 \to G_3 \leftarrow G_2$ be inclusions. The *intersection of* G_1 *and* G_2 is the hypergraph $G' = (E_1 \cap E_2, V_1 \cap V_2, \ell', \mathrm{cnct}')$ where $\ell'(e) = \ell_1(e)$ and $\mathrm{cnct}'(e) = \mathrm{cnct}_2(e)$ for all $e \in E_1 \cap E_2$. The *pullback of* $G_1 \to G_3 \leftarrow G_2$ is the pair of inclusions $G_1 \leftarrow G' \to G_2$ and the resulting square is a pullback square (see Figure 2.2).

Let $G_1 \leftarrow G_0 \to G_2$ be inclusions; they are *non-overlapping* if both $E_1 \cap E_2 \subseteq E_0$ and $V_1 \cap V_2 \subseteq V_0$ hold. The *pushout of non-overlapping inclusions* $G_1 \leftarrow G_0 \to G_2$ is the pair of inclusions $G_1 \to (G_1 +_{G_0} G_2) \leftarrow G_2$ where $(G_1 +_{G_0} G_2) = (E_1 \cup E_2, V_1 \cup V_2, \ell'', \mathrm{cnct}'')$ is the hypergraph such that

$$\ell''(e) = \begin{cases} \ell_1(e) & \text{if } e \in E_1 \\ \ell_2(e) & \text{if } e \in E_2 \end{cases} \text{ and } \mathrm{cnct}''(e) = \begin{cases} \mathrm{cnct}_1(e) & \text{if } e \in E_1 \\ \mathrm{cnct}_2(e) & \text{if } e \in E_2 \end{cases}$$

hold for all $e \in E_1 \cup E_2$; often, $(G_1 +_{G_0} G_2)$ is referred to as *the pushout of* G_1 *and* G_2 *over* G_0.

Without loss of generality, we shall assume that pairs of inclusions $G_1 \leftarrow G_0 \to G_2$ are always non-overlapping.

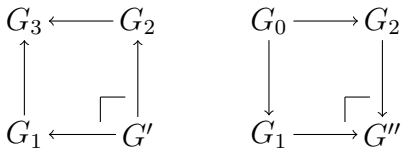
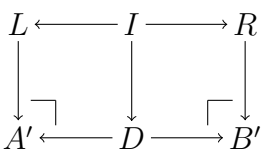


Figure 2.2: Pullback and pushout square

We are finally ready to introduce graph transformation systems and their "reduction" semantics.

Definition 2.2.5 (Rules and graph transformation systems)**.** A *rule* is a pair of inclusions of hypergraphs $\rho = (L \leftarrow I \to R)$. Let A, B be hypergraphs. Now, ρ *transforms* A *to* B if
there exists a diagram as shown to the right in which the two squares are pushouts, $A' \leftarrow I \to R$ is non-overlapping, and $A \cong A'$ and $B' \cong B$. A *graph transformation system* (GTS) is a pair $\mathcal{S} = (\Lambda, \mathcal{R})$ where Λ is a set of labels and $\mathcal{R}$ is a set of rules (over Λ-labeled hypergraphs).



A graph transformation rule can be understood as follows. Whenever the left hand side L is (isomorphic to) a sub-graph of some graph A then this sub-graph can be

"removed" from A, yielding the graph D. The vacant place in D is then "replaced" by the right hand side R of the rule. The middleman I is the memory of the connections that L had with the rest of the graph in order for R to be attached in exactly the same place. Also note, that if we remove a node in A, we have also to remove all incident edges explicitly, i.e. the deleted node has "the same" incident edges in L and A. Each graph transformation step can also be thought of as a chemical reaction according to the rule, which features as the reaction law. An example of a rewriting step is shown in Figure 2.3.

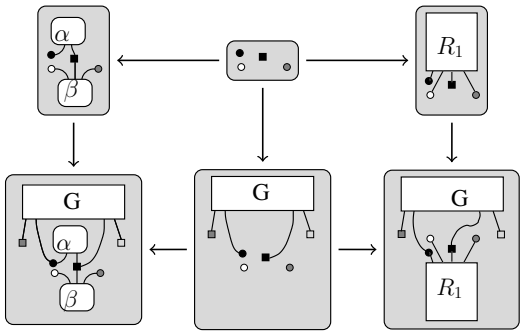


Figure 2.3: A rewriting step with rule "α/β".

As mentioned before, inclusions are given implicitly by the spatial arrangement of nodes and edges to keep the graphical representations clear.

As one might expect, the result of each transformation step is unique (up to isomorphism). This is a consequence of the following fact.

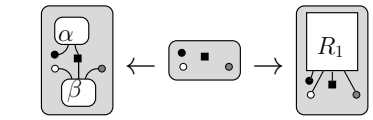
Fact 2.2.6 (Pushout complements)**.** *Let $G_2 \leftarrow G_1 \leftarrow G_0$ be a pair of hypergraph inclusions where $G_i = (E_i, V_i, \ell_i, \mathrm{cnct}_i)$ ($i \in \{0,1,2\}$) and assume that they satisfy the* dangling edge condition*: For all $v \in V_1 \setminus V_0$ there does not exist any edge $e \in E_2 \setminus E_1$ such that e is incident to v. Then there exists a unique sub-graph $G_2 \leftarrow D$ such that (2.1) is a pushout square.*

$$\begin{array}{ccc} G_1 & \longleftarrow & G_0 \\ \downarrow & \ulcorner & \downarrow \\ G_2 & \longleftarrow & D \end{array} \qquad (2.1)$$

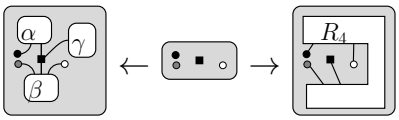
Definition 2.2.7 (Pushout Complement)**.** Let $G_2 \leftarrow G_1 \leftarrow G_0$ be a pair of hypergraph inclusions that satisfy the conditions of Fact 2.2.6; the unique completion $G_2 \leftarrow D \leftarrow G_0$ that yields the pushout square (2.1) is the *pushout complement* of $G_2 \leftarrow G_1 \leftarrow G_0$.

We now introduce the example graph transformation system that we shall use throughout the chapter to illustrate the basic ideas. The rule in the "reaction" in Figure 2.3 is part of this system.

Example 2.2.8 (Running Example). The system $\mathcal{S}_{ex} = (\Lambda, \mathcal{R})$ will be the following one in the sequel: The set of edge-labels is $\Lambda = \{\alpha, \beta, \gamma, \dots\}$ where $\mathtt{ar}(\alpha) = 2$, $\mathtt{ar}(\beta) = 3$ and $\mathtt{ar}(\gamma) = 1$; moreover, $\mathcal{R}$ is the set of rules given in Figure 2.4 where the R_i represent different graphs (e.g. edges with labels R_i).



(a) Rule "α/β"



(b) Rule "$\alpha/\beta/\gamma$"

Figure 2.4: Reaction rules of $\mathcal{S}_{ex}$.

In this subsection, we have presented the double pushout approach as a model for "reactions" that occur in a system about which one has complete knowledge. Thus rewriting is similar to the reaction semantics for CCS. Now, we come to the more recent and central idea that graph transformation systems "automatically" have an interactive nature, which endows each graph with a behavior. This is similar to process calculi where process terms cannot only react but also exhibit behavior that depends on possible interactions with other processes.

2.2.3 Behavior as interaction with the environment

We now explain how to use the Borrowed Context technique [19] to equip each graph transformation system with a labeled transition semantics that models the interactive behavior of systems that are specified as graphs with graph transformation rules. Each labeled transition will model an interaction of a graph with an "external" environment; in the the world of process calculi, this environment is formalized as an arbitrary (reactive) context.

Intuitively, one might want to "hide" parts of a graph from the environment while some portion of the state is directly exposed. Thus, each state of the labeled transition system (LTS) will be a graph with an interface, which makes part of the graph directly accessible whereas the remainder is "hidden"; more technically, the interface is also needed to have a meaningful way to consider the graph within an "external" context. To avoid confusion, we emphasize here that the labels of transitions in the LTS will depend directly on the rules of the GTS (and thus only indirectly on the edge-labels Λ). We use the standard definition of labeled transition systems, which we recall here to fix notation.

Definition 2.2.9 (Labeled transition system)**.** A *labeled transition system* (LTS) is a tuple (S, Ξ, R) where S is a set of *states*, Ξ is a set of *labels* and $R \subseteq S \times \Xi \times S$ is the *transition relation*. We write

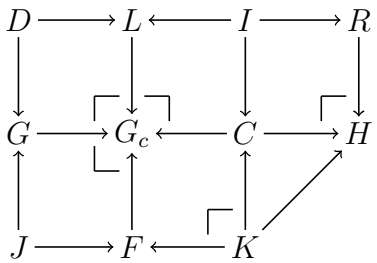
$$s \xrightarrow{a} s'$$

if $(s, a, s') \in R$ and say that s can evolve to s' by performing a.

Before we delve into the technical details of the LTS semantics for graphs, we first discuss the main ideas informally. The states will be graphs with interface $J \to G$. The "larger" part G models the whole "internal" state of the system while the "smaller" part, the interface J, models the part that is directly accessible to the environment and allows for (non-trivial) interaction. As a particularly simple example, one could have a Petri net where the set of places (with markings) is the complete state and some of the places are "open" to the environment such that interaction takes place by exchange of tokens.

More generally, the addition of agents/resources from the environment to (the interface of) a state might result in "new" reactions, which have not been possible before. In the Petri net scenario, extra tokens might enable transitions that could not be fired before. The idea of the LTS semantics for graph transformation is to consider as labels "minimal" contexts that trigger "new" reactions (by providing extra agents/resources). For a more formal treatment of this intuition, see [56]. A direct definition of the LTS semantics for graph transformation can be given in terms of so-called borrowed context diagrams.

Definition 2.2.10 (DPOBC)**.** Let $\mathcal{S} = (\Lambda, \mathcal{R})$ be a graph transformation system. Its LTS has all inclusions of hypergraphs $J \to G$ as *states* where J is called the *interface*. Labels are pairs of inclusions $J \to F \leftarrow K$. A state $J \to G$ evolves to another one $K \to H$ if there is a diagram as shown to the right, which is called a DPOBC-*diagram* or just a BC-*diagram*: All morphisms are injective and the squares are pullbacks or pushouts as marked. In such a BC-diagram, $G \leftarrow D \to L$ is called the *partial match of L (in G)*.



We often speak of the graph D as the partial match for a transition; in fact, the whole BC-diagram is determined by the rule and the partial match (up to isomorphism).

Example 2.2.11. An example of a DPOBC-diagram is given in Figure 2.5. The original state $J \to G$ contains a hyperedge α. The partial match D is exactly this edge. The state can therefore evolve using rule α/β, provided that it borrows β from the environment.

The bigger graph G_c contains G but is also completed with what is missing such that the left-hand side of the rule can be embedded into it. It can thus be rewritten, as shown in Figure 2.3. The last row shows the "evolution of the interface during the rewriting": First, we have the original interface; then the missing part is added; finally, some elements from the interface have to be removed if they have been deleted

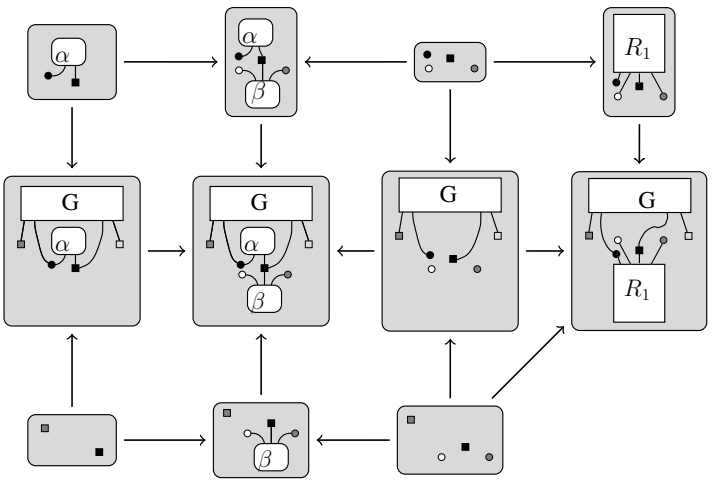


Figure 2.5: An example of a BC-diagram that uses the rule "α/β".

by application of the rule.

That the label $J \to F \leftarrow K$ in Definition 2.2.10 is "minimal" is captured by the two leftmost squares in the BC diagram above: The "addition" $J \to F$ is "just enough" to complete the partial match of the left hand side of the rule $L \leftarrow I \to R$. That the interaction with the environment involves a reaction is captured by the other two squares in the upper row in the BC-diagram. During this reaction, some agents might disappear or some resources might be used (depending on the preferred metaphor) and new ones might come into play. Finally the bottom left pullback square in the BC-diagram restricts the changes to obtain the new interface into the resulting state.

Different rules might result in different deletion effects that are "visible" to the environment. Thus, the full label of each such "new" reaction is the "trigger" $J \to F$ together with the "observable" change $F \leftarrow K$ (with state $K \to H$ after interaction). Note that more recent process calculi also have several reaction rules (as for example the ambient calculus) while CCS has only a single one.

2.3 A process calculus perspective on borrowed contexts

We only assume familiarity with process calculi and in particular do not require knowledge of the Borrowed Context technique (beyond the definition in the previous section). This section should also clarify the purpose and relevance of the main results in Section 2.4 and Section 2.5.

We begin with an informal motivation by developing an analogy with the axioms

and rules of CCS. The axioms will provide *basic actions*, which can be seen as a generalization of transitions of the form $\alpha.P \xrightarrow{\alpha} P$ in CCS. After a quick review of how contexts are formalized in graph transformation, we shall give rules that allow to "embed" transitions into contexts; these rules are similar to the rule that allows to infer $P \parallel Q \xrightarrow{\alpha} P' \parallel Q$ from $P \xrightarrow{\alpha} P'$ in CCS because $[\cdot] \parallel Q$ is a "non-interfering" context. Finally, we show formally that axioms for basic actions with two "contextualization" rules exactly capture the Borrowed Context technique.

2.3.1 The analogy with CCS

The axioms of our system will be similar to the axioms of CCS, where the process $\alpha.P$ can perform the action α and then behaves as P; this is usually written $\alpha.P \xrightarrow{\alpha} P$ where α ranges over the actions $a, \overline{a}$ (and τ). In particular, a and $\overline{a}$ are co-actions of each other, which are consumed during the reaction of $a.P$ and $\overline{a}.P$ in the combined process $a.P$ and $\overline{a}.P$.

In the case of graphs, each rule $L \leftarrow I \rightarrow R$ gives rise to a whole family of such actions – one for each subgraph of L. More precisely, each subgraph D of L can be seen as an "action"; each such action has a co-action $\widehat{D}^L \rightarrow L$ such that L is the union of D and $\widehat{D}^L$ (and $\widehat{D}^L$ is the minimal sub-graph with this property). For example, in the rule α/β, both edges α and β yield (complementary) basic actions. Indeed, to make the analogy closer, the common node between the two edges in the left hand side of the rule α/β is the analogue of a channel; one of the edges performs the input and the other the output "on" the common node.

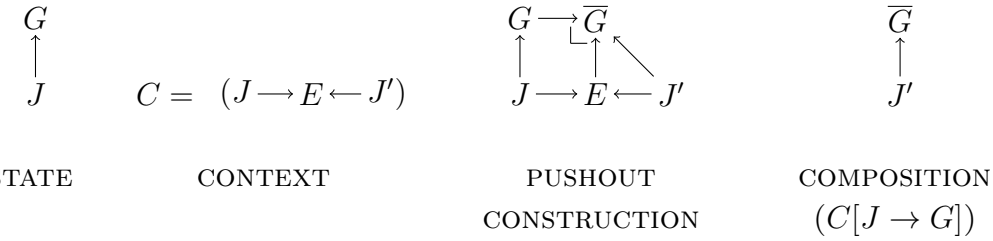
Formally, in Table 2.1, we have the family of *Basic Action* axioms. It essentially represents all the possible uses of a transformation rule. In (an encoding of) CCS, the left hand side would be a pair of unary edges a and $\overline{a}$, which both disappear during reaction. Now, if only a is present "within" the system, it needs $\overline{a}$ to perform a reaction; thus, the part a of the left hand side induces the (inter-)action that consists in "borrowing" $\overline{a}$ and deleting both edges (and similarly for $\overline{a}$). In general, e.g. in the rule $\alpha/\beta/\gamma$, there might be more than two edges that are involved in a reaction and thus we have a whole family of actions. More precisely, each portion of a left hand side induces the action that consists in borrowing the missing part to perform the reaction (thus obtaining the complete left hand side), followed by applying the changes that are described by the rule.

Next, we shall describe counterparts for the two CCS-rules that allow to perform a given action in parallel to another process and under a restriction; the respective forms of contexts in which actions can be performed are "parallel contexts" $[\cdot] \parallel Q$ and "restriction contexts" $(\nu b)[\cdot]$. More precisely, whenever we have the transition $P \xrightarrow{\alpha} P'$ in CCS and another process Q, then there is also a transition $P \parallel Q \xrightarrow{\alpha} P' \parallel Q$; similarly, we also have $(\nu b)P \xrightarrow{\alpha} (\nu b)P'$ whenever $\alpha \notin \{\overline{b}, b\}$. More abstractly, actions are preserved by certain contexts and not by others; for example $a.[\cdot]$ does block all

actions.

In the case of graph transformation, there is a natural counterpart for process contexts $C[\cdot]$ such as $P \parallel [\cdot]$ and $(\nu b)[\cdot]$. The only complication is that graphs have arbitrary interfaces $J \to G$ (see also Definition 2.2.10) while processes have a sacrosanct "interface", viz. their free names. Thus, graph contexts have a "type", which is an interface graph J; only states with interface J can be put into a context of this "type". The result is called the composition[1] of the state with the context.

Definition 2.3.1 (Context and composition)**.** Let $J \to G$ be a state. A *context (of type J)* is a pair of inclusions $C = J \to E \leftarrow J'$. The *composition* of $J \to G$ with the context C, written $C[J \to G]$, is the inclusion of J' into the pushout of $E \leftarrow J \to G$ as illustrated in the following figure (with the assumption that C is free for $J \to G$).



The left inclusion of the context, i.e. $J \to E$ in the definition, can also be seen as a state with the same interface. The pushout then gives the result of "gluing" E to the original G at its interface J; the second inclusion $J' \to E$ models a new interface, which possibly contains part of J and additional "new" entities in E.

The idea of our stratified presentation of the Borrowed Context technique is based on the observation that each BC-transition

$$(J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H),$$

which might be a basic action or not, remains roughly unchanged in contexts of a certain form; in other words, some contexts C allow $C[J \to G]$ to perform "the same" action as $J \to G$.

With this observation in mind, we shall first characterize two classes of contexts that are *non-interfering* in the described sense. These two classes roughly correspond to CCS contexts of the form $P \parallel [\cdot]$ and $(\nu b)[\cdot]$. However, even though "non-interfering" contexts have no substantial influence on actions, we will have to keep track on what they add and how they change interfaces. Finally, at the end of this section, we show that axioms for basic actions together with the two natural contextualization rules for non-interfering contexts yield a sound a complete description of the Borrowed Context technique.

[1] The reason for this is that the construction in Definition 2.3.1 is essentially the composition of co-spans.

2.3.2 Borrowed contexts in three layers

In this subsection we shall provide the formal details of a process calculus like presentation of the Borrowed Context technique. We have already discussed the idea of basic actions and non-interfering contexts. We fix now a graph transformation system $\mathcal{S} = (\Lambda, \mathcal{R})$ and – relative to this parameter – define the system 3L.

Basic Actions We start with the axioms of the system. They derive the basic actions as discussed above; and example of a basic action is given in Figure 2.6.

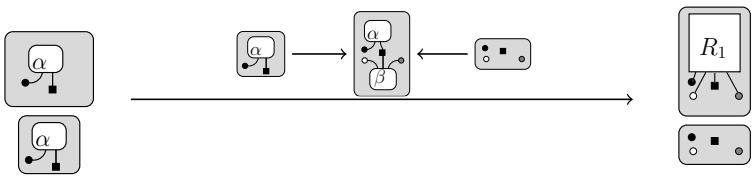


Figure 2.6: An example of a basic action.

Definition 2.3.2 (Basic Action axioms)**.** Let $(L \leftarrow I \rightarrow R) \in \mathcal{R}$ be a rule and let $D \rightarrow L$ be a sub-graph. Then

$$\frac{}{(D \rightarrow D) \xrightarrow{D \rightarrow L \leftarrow I} (I \rightarrow R)}$$

is a *Basic Action Axiom* of 3L.

Interface Narrowing Next, we address the counterpart of name restriction. This means, we first define the counterpart of CCS-contexts of the form $(\nu a)[\cdot]$; these are just contexts of the form $C = J \rightarrow J \leftarrow J'$, which will be called *narrowing contexts*. Intuitively, such a context does not interfere with a transition with label $J \rightarrow F \leftarrow K$ if J' is still big enough to glue all new entities in F. This is in direct analogy to CCS, where the restriction (νa) preserves only those actions that do not involve a. While we do not have to adjust labels in CCS, even non-interfering narrowing context "narrows" the label of the transition while the "proper" action remains untouched. This is made formal in the following definition and the analogy to CCS is made more precise afterwards.

Definition 2.3.3 (Narrowing)**.** A *narrowing context* is a pair of inclusions $C = J \rightarrow J \leftarrow J'$ in which only the right inclusion might be proper. Let $J \rightarrow F \leftarrow K$ be a label. The narrowing context $C = J \rightarrow J \leftarrow J'$ does *not interfere* with the label if the pushout complement of $F \leftarrow J \leftarrow J'$ exists. If C is non-interfering, then the *C-narrowing* of the label, written $C\langle J \rightarrow F \leftarrow K\rangle$, is the lower row in the following figure

$$C\langle J \to F \leftarrow K\rangle := \begin{array}{ccccc} J & \longrightarrow & F & \longleftarrow & K \\ \uparrow & & \uparrow & & \uparrow \\ J' & \longrightarrow & F' & \longleftarrow & K' \end{array} \qquad \text{where } C = J \to J \leftarrow J'$$

where the left square is a pushout and the right one a pullback. Whenever we write $C\langle J \to F \leftarrow K\rangle$, we assume that the relevant pushout complement exists.

If we think of the interface as the set of free names of a process, then restricting a name means removing it from the interface. Thus, J' plays the role of the set of all remaining free names. If the pushout complement F' exists, it represents F with the restricted names erased. Finally, since a pullback here can be seen as an intersection, K' is K without the restricted names. So we finally obtain the "same" label where "irrelevant" names are not mentioned. It is of course not always possible to narrow the interface. For instance, one cannot restrict the names that are involved in labeled transitions of CCS-like process calculi. This impossibility is captured by the non-existence of the pushout complement.

Narrowing contexts just make interfaces smaller; the remainder of the involved states is left unchanged. An example of interface narrowing is given in Figure 2.7

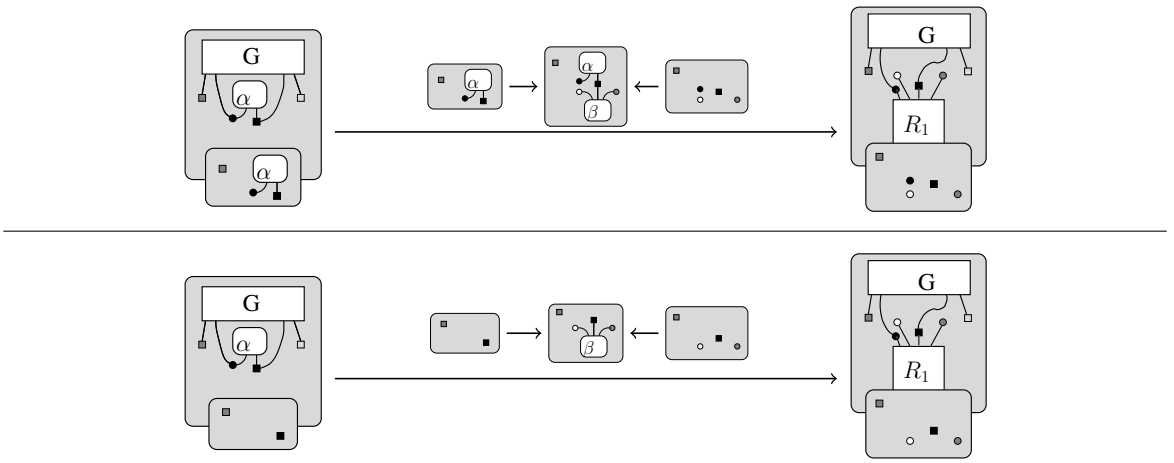


Figure 2.7: An example of interface narrowing

where the narrowing context removes the α-labeled edge and the first of its nodes from the interfaces. Interface narrowing yields the first rule scheme in the system 3L.

Definition 2.3.4 (Narrowing rule)**.** Let $J \to F \leftarrow K$ be a label, let $C = J \to J \leftarrow J'$ be a non-interfering narrowing context, and let $J' \to F' \leftarrow K' = C\langle J \to F \leftarrow K\rangle$ be the C-narrowing of the label; moreover let $J \to G$ and $K \to H$ be inclusions. Then

$$\frac{(J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)}{(J' \to G) \xrightarrow{J' \to F' \leftarrow K'} (K' \to H)}$$

is an instance of the narrowing rule of 3L.

Compatible contexts It remains to define contexts that correspond to parallel composition with another process P in CCS. In the case of graph transformation, this case is slightly more involved than one might expect. The problem is that even the "pure" addition of context potentially interferes with transitions. For example, if an interaction involves the deletion of an (isolated) node in the interface, the addition of an edge to this node blocks the reaction. Thus a context that only adds new entities, which will be called *monotone*, interferes if it creates dangling edges. Non-interfering, monotone contexts are intuitively similar to CCS-contexts of the form $P \parallel [\cdot]$; they are called *compatible*.

Definition 2.3.5 (Compatible contexts)**.** Let $C = J \to E \leftarrow \overline{J}$ be a context; it is *monotone* if $J \to \overline{J}$. Let $J \to F \leftarrow K$ be a label; now C does not *interfere with* $J \to F \leftarrow K$ if it is possible to construct the diagram in (2.3.2) where both squares are pushouts. Finally, a context $J \to E \leftarrow \overline{J}$ is *compatible* with the label $J \to F \leftarrow K$ if it is monotone and does not interfere with the label.

$$\begin{array}{ccccc} J & \longrightarrow & F & \longleftarrow & K \\ \downarrow & & \downarrow & & \downarrow \\ E & \longrightarrow & E_1 & \longleftarrow & E' \end{array}$$

In a label $J \to F \leftarrow K$, the left inclusion represents the addition of new entities that "trigger" a certain reaction. A compatible context is simply a context that preserves the old interface and adds new entities that do not block the reaction, i.e. it does not add new edges to nodes that disappear during the interaction. An illustration of how to embed of a whole transition into a monotone context is given in Figure 2.8.

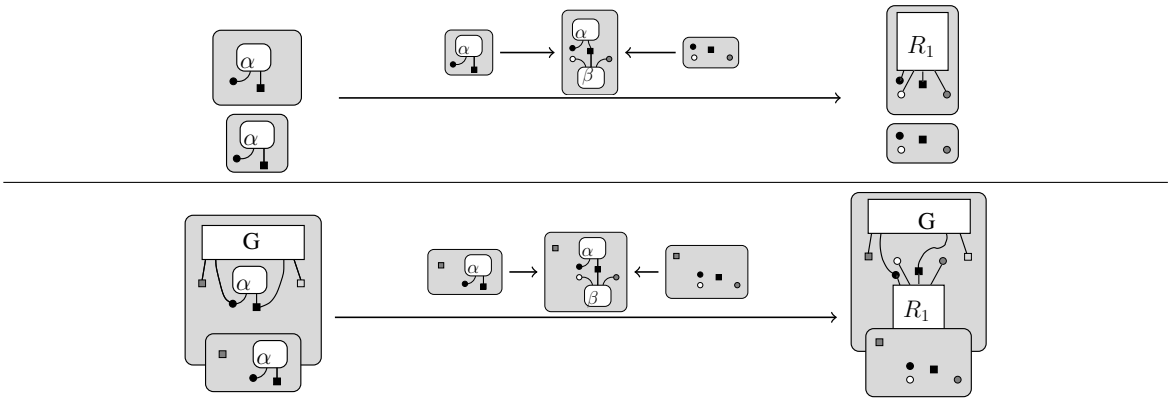
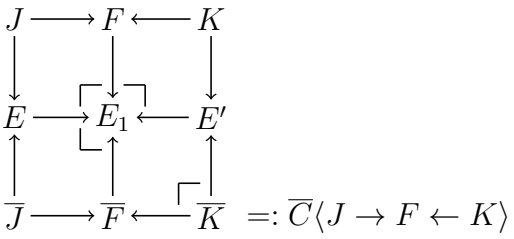


Figure 2.8: A transition in a monotone context.

To properly define a rule for monotone contexts, we introduce a partial operation for the *combination* of co-spans with a common interface, which generalizes the narrowing construction.

Definition 2.3.6 (Cospan combination)**.** Let $C = (J \to F \leftarrow K)$ and $\overline{C} = (J \to E \leftarrow \overline{J})$ be two co-spans. They are *combinable* if there exists a diagram of the following form.



The label $\overline{J} \to \overline{F} \leftarrow \overline{K}$ is the *combination of C with $\overline{C}$*, and is denoted by $\overline{C}\langle J \to F \leftarrow K\rangle$.

In fact, it is easy to show that compatible contexts are combinable with their label.

Lemma 2.3.7. *Given a label $J \to F \leftarrow K$ and a compatible context $J \to E \leftarrow \overline{J}$, we can split the diagram in (2.3.2) to obtain the following diagram.*

With this lemma we can easily define the rule that corresponds to "parallel composition" of an action with another "process".

Definition 2.3.8 (Compatible Contexts rule)**.** Let $J \to F \leftarrow K$ be a label, let $C = J \to E \leftarrow \overline{J}$ be a context that is compatible with it, and let $\overline{C} = (J \to F \leftarrow K)\langle C\rangle$ be the combination of C with the label; moreover let $J \to G$ and $K \to H$ be inclusions. Then

$$\frac{(J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)}{C[J \to G] \xrightarrow{C\langle J \to F \leftarrow K\rangle} \overline{C}[K \to H]}$$

is an instance of the combination rule in 3L.

Soundness and Completeness The 3L-system, which consists of basic actions, the narrowing rule and the combination rule is summarized in Table 2.1. It does not only give an analogy to the standard SOS-semantics for CCS. In fact, we shall see that the labels that are derived by the standard BC technique are exactly those labels that can be obtained from the basic actions by compatible contextualization and interface narrowing. In technical terms, the 3L-system is sound and complete.

Theorem 2.3.9 (Soundness and completeness)**.** *Let $\mathcal{S}$ be a graph transformation system. Then there is a* BC*-transition*

$$(J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)$$

- **Basic Actions**

$$\frac{}{(D \to D) \xrightarrow{D \to L \leftarrow I} (I \to R)} \quad \text{where} \quad (L \leftarrow I \to R) \in \mathcal{R} \text{ and } D \to L$$

- **Interface Narrowing**

$$\frac{(J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)}{(J' \to G) \xrightarrow{J' \to F' \leftarrow K'} (K' \to H)} \quad \text{where} \quad C = J \to J \leftarrow J' \text{ and } J' \to F' \leftarrow K' = C[J \to F \leftarrow K]$$

- **Compatible Contextualization**

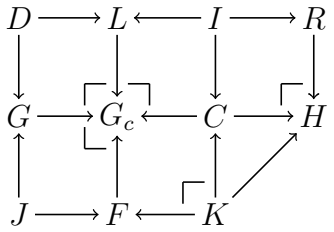
$$\frac{(J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)}{C[J \to G] \xrightarrow{C[J \to F \leftarrow K]} \overline{C}[K \to H]} \quad \text{where} \quad C = J \to E \leftarrow \overline{J} \text{ is compatible with } J \to F \leftarrow K \text{ and } \overline{C} = (J \to F \leftarrow K)[C]$$

Table 2.1: Axioms and rules of the 3L-semantics.

if and only if it is derivable in the 3L*-system.*

Proof sketch. It is easy to build a DPOBC-diagram to justify the Basic Action axioms. We have seen while defining the Narrowing and Compatible Context rules that they are derivable with the BC technique. Let us now show completeness.

Let d be a DPOBC-diagram using the rule $\rho = L \leftarrow I \to R$; the resulting transition is $t = (J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)$ and the partial match is D.



We have that the transition $t_0 = (D \to D) \xrightarrow{D \to L \leftarrow I} (I \to R)$ is a basic action, i.e. derivable by an axiom of the system. Let $C = D \to G \leftarrow G$ be a monotone context. It is clearly non-inhibiting w.r.t. $D \to L \leftarrow I$. Thus it is compatible with it and $\overline{C}\langle D \to L \leftarrow I \rangle = G \to G_c \leftarrow C$. So one can use the contextualization rule, and obtain, from t_0, the transition

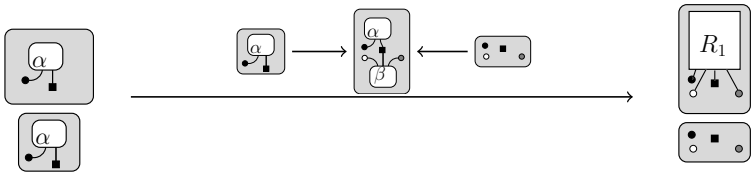
$$t'' = (G \to G) \xrightarrow{G \to G_c \leftarrow C} (C \to H).$$

Let $C' = G \to G \leftarrow J$. By uniqueness of pushout complement and pullback, the C-narrowing of $G \to G_c \leftarrow C$ is exactly $J \to F \leftarrow K$. Therefore the narrowing rule applied to t'' yields the original transition t. □

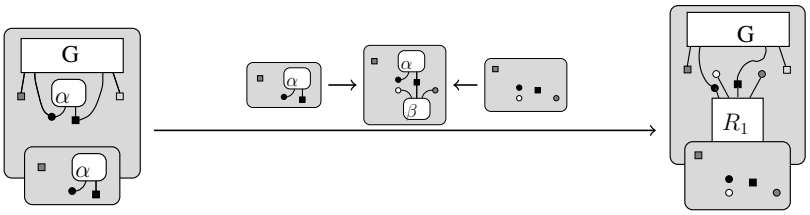
As a result, for any DPOBC-diagram that justifies a transition t (where the names of the graphs are the usual ones, as in Definition 2.2.10), the following three step derivation in 3L justifies t:

$$\dfrac{\dfrac{\dfrac{}{(D \to D) \xrightarrow{D \to L \leftarrow I} (I \to R)}\,\texttt{ax.}}{(G \to G) \xrightarrow{G \to G_c \leftarrow C} (C \to H)}\,\texttt{ctx.}}{(J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)}\,\texttt{narr.}$$

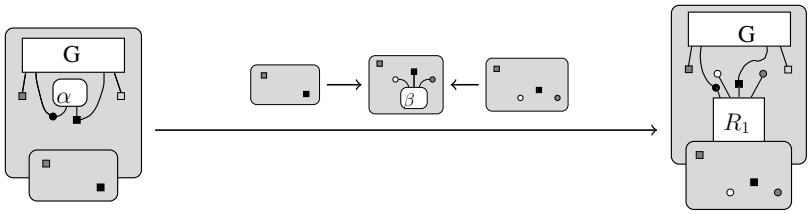
Example 2.3.10. An example of a derivation of the example transition is shown in Figure 2.9. We can see the basic action first, using the partial match of the diagram. Using compatible contextualization, we "add" to this transition, all that is in the original state. The interface is everything that is necessary; in this example, we just add an extra vertex (and we could have put some more objects in the interface of the monotone context). Finally, we remove from the interface everything that is not needed, to get the desired interface.



(a) First, make the partial match evolve (Basic Action).



(b) Then, add what is missing from the original graph, thanks to Compatible Context.



(c) Finally, remove the interface you don't need, using Interface Narrowing.

Figure 2.9: An example of a derivation in the 3L-system.

The main role of soundness and completeness is not its technical "backbone", which is similar to many other theorems on the Borrowed Context technique. The

main insight to be gained is the absence of any "real" communication between subsystems; roughly, every reaction of a state can be "localized" and then derived from a basic action (followed by contextualization and narrowing). In particular, we do not have any counterpart to the communication-rule in CCS, which has complementary actions $P \xrightarrow{a} P'$ and $Q \xrightarrow{\overline{a}} Q'$ as premises and allows to infer communication of the processes P and Q, i.e. a silent "internal" transition $P \parallel Q \xrightarrow{\tau} P' \parallel Q'$. This absence of communication in the "monolithic" BC-labels is the main motivation for our study of composition of transitions.

2.4 Communication in composed states

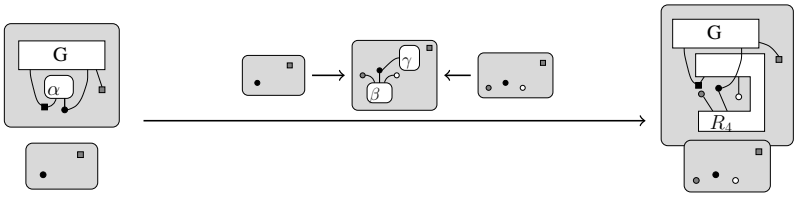
The formal analogy between CCS and the Borrowed Context technique that we have established in the previous section is imperfect: we have no counterpart to the communication rule of CCS, which allows to derive the reaction $P \parallel Q \xrightarrow{\tau} P' \parallel Q'$ of the "composed" state $P \parallel Q$ from the two interactions $P \xrightarrow{a} P'$ and $Q \xrightarrow{\overline{a}} Q'$ of the "constituents" P and Q. Thus, we shall now analyze when and how two labeled transitions from two different states in a GTS give rise to a "smaller" labeled transition of the composition of the two states. This analysis will lead to a counterpart of the communication rule of CCS that is admissible in the system 3L; moreover, as we shall exploit in Section 2.5, we can reduce the number of axioms.

2.4.1 The idea of composition of transitions

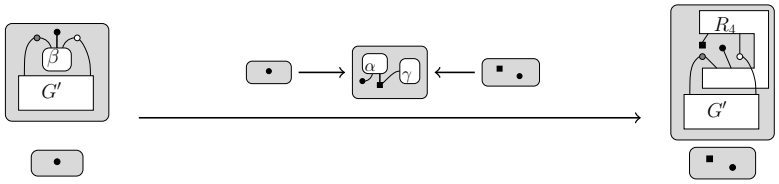
Communication within a composed state is based on the following idea: (sub-)states may provide resources for each other that they (used to) borrow from the environment; as a consequence, the composed state needs to borrow less from the environment. This idea is illustrated in the following example.

Example 2.4.1 (Composition of transitions). Let $s = J \to G$ be a state of $\mathcal{S}_{ex}$ that contains an edge α with its second connected node in the interface as shown in Figure 2.10(a). Further, let $s' = J' \to G'$ be a state that contains an edge β with its second connected node in the interface as shown in Figure 2.10(b). Both graphs can perform transitions t and t', using the rule $\alpha/\beta/\gamma$. Let X be the graph that consist of the black round vertex only, which is both a subgraph of J and J'. Now we can compose t and t' along $J \leftarrow X \to J'$ to obtain the transition in Figure 2.10(c).

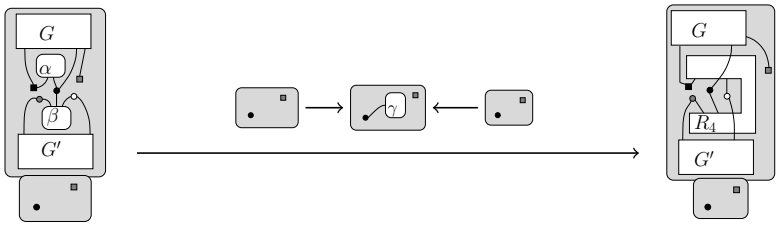
The crucial problem of a general composition rule for the system 3L is due to the inherent, "incomplete" information of labels. The transitions that we derive with the Borrowed Context technique only indicate what a state needs from the environment to perform *some* reaction that the state cannot perform on its own; in particular the transition label abstracts away from the complete BC diagram. Roughly, labels indicate what needs to "be around" to make something happen but do not inform



(a) A first transition for a state $s = J \to G$.



(b) A second transition for a state $s' = J' \to G'$.



(c) The composition of s and s' along $J \leftarrow X \to J'$.

Figure 2.10: An example of composition of two transitions that use the rule $\alpha/\beta/\gamma$.

about what exactly is happening "inside" the interacting state. For instance, a state can react after "borrowing" some small graph F; however the graph F could be used in several ways – possibly even applying different rules.

Thus, in general, one can neither determine what part of a state is actually reacting, i.e. what partial match has been used to derive the label, nor what rule is used. Thus, it is non-trivial to generalize the communication rule of CCS to a composition rule for "opposite labels" – simply, because *the* "opposite" of a derived label does not exist. The following example illustrates the problem and also suggests that it is not due to the use of graphs as system models but is rather a consequence of the use of minimal contexts as labels.

Example 2.4.2 (Failure of Composition). Consider the following microscopic (process) calculus. The terms are given by

$$P ::= ?a \mid !a \mid \flat a \mid \natural a \mid 0 \mid P \parallel P$$

where a is element of a set of names and we have $\parallel$ as an associative operator. Moreover, we have the following reaction axioms and rules for contextualization.

$$?a \parallel !a \to 0 \qquad \flat a \parallel !a \to 0 \qquad ?a \parallel \natural a \to 0$$

$$\frac{P \to Q}{P \parallel R \to Q \parallel R} \qquad \frac{P \to Q}{R \parallel P \to R \parallel Q}$$

With the intuitive idea of minimal contexts as labels, we have the two labeled transitions $?a \xrightarrow{[\cdot \parallel !a]} 0$ and $!a \xrightarrow{[?a \parallel \cdot]} 0$; they can be composed, leading to the "silent" transition $?a \parallel !a \xrightarrow{[\cdot]} 0$. However, we also have $\flat a \xrightarrow{[\cdot \parallel !a]} 0$ and $\natural a \xrightarrow{[?a \parallel \cdot]} 0$ but not $\flat a \parallel \natural a \xrightarrow{[\cdot]} 0$. Thus, two transitions with labels $[\cdot \parallel !a]$ and $[?a \parallel \cdot]$ are not always composable. This means that composability depends on the states. Nevertheless, there is at most one way to compose "opposite" labels.

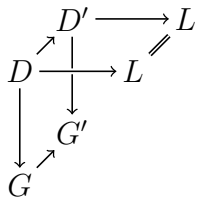
In the case of graphs, the use of the Borrow Context technique implies the existence of some "substate" that is "responsible" for an interaction with the environment. Moreover, in the composition of the two labeled transitions in Example 2.4.1, we can still locate the "responsible" substates of the constituents, namely the edges with labels α and β. Finally, we see that the new "responsible" substate is the union of the old ones. Thus, the interaction of the composed state is not only based on the same rule, but it actually reuses the same parts of the original states that triggered the transition.

2.4.2 Composition results for Borrowed Context diagrams

We now formalize the idea that labeled transitions of "composed" states can be "restricted" to interactions of their "constituents". We start by defining superstates of states (which in turn will be substates) and formally describe how transitions of substates can be extended to superstates.

Definition 2.4.3 (Superstate and homogeneous transitions)**.** Let $s = J \to G$ and $s' = J' \to G'$ be two states. Now s' is a *superstate* of s and s is a *substate* of s' if $s' = C[s]$ for some monotone context $C = J \to F \leftarrow J'$ (see Definition 2.3.5).

Let $\rho = L \leftarrow I \to R$ be a rule, let $J' \to G'$ be a superstate of $J \to G$ and let $t = (J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)$ and $t' = (J' \to G') \xrightarrow{J' \to F' \leftarrow K'} (K' \to H')$ be two transitions that are derived with respective partial matches D and D'. Now, t' is *homogeneous* with respect to t and t *extends to* t' *(relative to* D *and* D'*)* if $D \to D'$.



The extension of a transition yields the illustrated diagram (as part of the complete BC diagrams).

Now we are ready to formulate our composition results: in Proposition 2.4.4, we describe how two transitions that are derived using partial matches into a common rule can be composed such that the original states have a "minimal" overlap in the composed state, namely the intersection of the partial matches; further, Theorem 2.4.6 gives sufficient conditions for the composition of two transitions such that the resulting transition is homogeneous w.r.t. to both of them (relative to their partial matches).

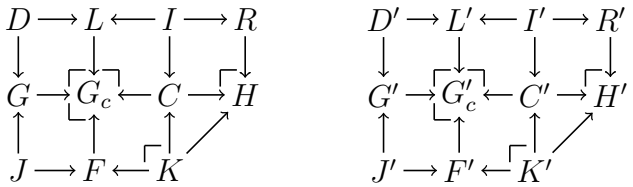
Composition with minimal overlap Homogeneity expresses the fact that the rule is applied in the superstate exactly where we expect it to be, i.e. its partial match "reuses" the elements that were already in the partial match in the original graph. In the opposite direction, for any transition $\bar{t}$ from a state $\overline{s} = \overline{J} \to \overline{G}$ that is derived using a partial match $\overline{D}$, any substate $s = J \to G$ of $\overline{s}$ can evolve in "the same" way by "restricting" $\bar{t}$ to a transition t from s such that $\bar{t}$ is homogeneous w.r.t. t: we simply take the intersection of G and $\overline{D}$ as partial match for t. In general, s might miss some parts that were in $\overline{s}$ to evolve and thus the superstate $\overline{s}$ will need to borrow less from the environment.

Now consider states s and s' with respective transitions t and t' that are derived with respective partial matches D and D' into a common rule $\rho = L \leftarrow I \to R$. Suppose we want to extend t and t' to a composed transition $\bar{t}$ such that it needs to borrow as little as possible from the environment (relative to D and D'). In fact, the solution uses the union of D and D' as partial match and yields a minimal overlap of s and s' in the composed state $\overline{s}$.

Proposition 2.4.4 (Composition with minimal overlap)**.** *Let $s = J \to G$ and $s' = J' \to G'$ be states, let $\rho = L \leftarrow I \to R$ be a rule, and let $t = (J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)$ and $t' = (J' \to G') \xrightarrow{J' \to F' \leftarrow K'} (K' \to H')$ be two transitions that are derived using respective partial matches D and D' into L.*

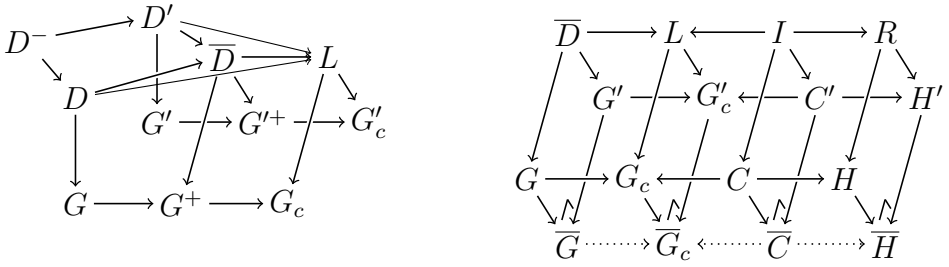
Then there exists a transition $\bar{t} = (\overline{J} \to \overline{G}) \xrightarrow{\overline{J} \to \overline{F} \leftarrow \overline{K}} (\overline{K} \to \overline{H})$ that is homogeneous w.r.t. both t and t' (relative to D and D') where $G \to \overline{G} \leftarrow G'$ is the pushout of $G \leftarrow (D \cap D') \to G'$ and $\overline{J} = J \cup J'$.

Proof. Let d and d' be the DPOBC-diagrams yielding t and t' using rule ρ, with partial matches D and D' where we use the usual names for objects in DPOBC diagrams (as in Definition 2.2.10).



We shall build a DPOBC-diagrams $\overline{d}$ using rule ρ and yielding a transition $\bar{t}$ where $\overline{J} \to \overline{G}$ is as in the statement of the proposition. The outline of the proof is as follows: we start by constructing the graph $\overline{G}$ and the first row of the BC diagram $\overline{d}$, then we build the interface $\overline{J}$, and finally we show that there exists a pushout complement for $\overline{J} \to \overline{G} \to \overline{G}_c$.

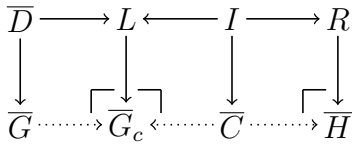
Let D^- be the pullback of $D \to L \leftarrow D'$ and let $\overline{D}$ be the union of D and D' over L, i.e. $D \to \overline{D} \leftarrow D'$ is the pushout of $D \leftarrow D^- \to D'$; now we use $\overline{D}$ to split the upper left pushout squares of d and d' as shown in Figure 2.11(a).



(a) (b)

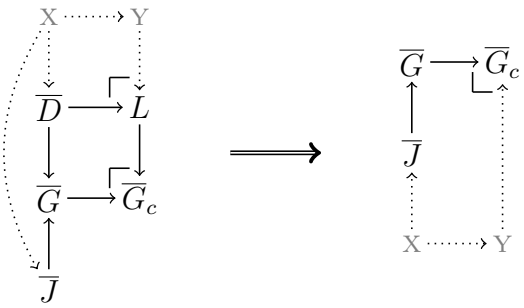
Figure 2.11

Next, take all pushouts and mediating morphisms as shown in Figure 2.11(b). It is straightforward to verify that all faces are pushouts: by composition of pushout squares, the vertical "diagonal" squares are pushouts. We have thus constructed the first row of $\overline{d}$.



As $\overline{J}$, let us take the union of J and J' in $\overline{G}$. It is clear that $\overline{J} \to \overline{G}$ is a superstate of $J \to G$ and $\overline{J}' \to \overline{G}'$.

Let us show now that a pushout complement for $\overline{J} \to \overline{G} \to \overline{G}_c$ exists. It is sufficient to show that there exist graphs X and Y such that Y is a pushout complement of $X \to \overline{D} \to L$ and $X \to \overline{J}$, as shown in the diagram below.



In d and d' we already have pushout complements for D and D' in L as shown in Figure 2.12(a). By splitting the back pushout squares through $\overline{D}$ and constructing the pullbacks in Figures 2.12(b) and 2.12(c), we obtain a pushout square where Y is the pushout complement of $X \to \overline{D} \to L$. It is now sufficient to show that $X \to \overline{J}$, which is a consequence of the following reasoning in the distributive lattice of subgraphs.

$$X = X \cap \overline{D} = X \cap (D \cup D') = (X \cap D) \cup (X \cap D') \subseteq E \cup E' \subseteq J \cup J' = \overline{J} \quad (2.2)$$

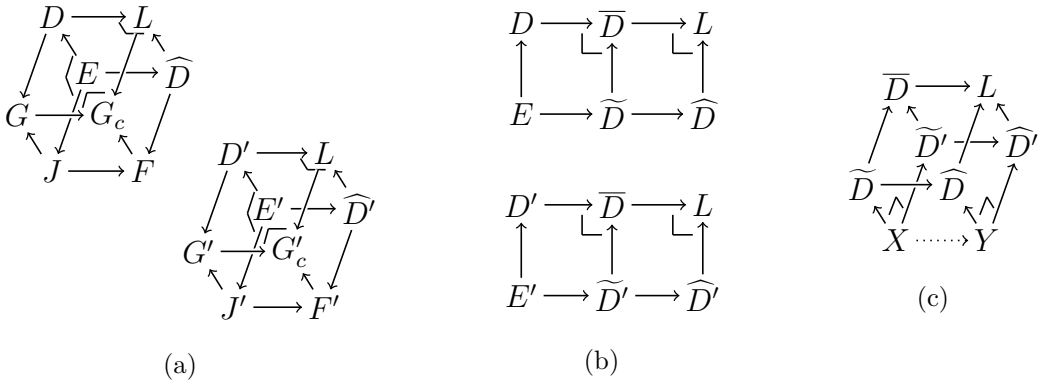


Figure 2.12

□

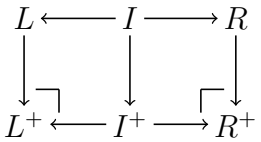
This proposition already generalizes the preliminary results that have been described in [15]. However, it is still too specialized to obtain a meaningful counterpart to the communication rule of CCS; nevertheless, it can be "re-used" to obtain the main theorem about the composition of DPOBC-diagrams.

Composition with arbitrary overlap The composition for states that is used in Proposition 2.4.4 uses as default a minimal overlap of the substates in the composed super-state; in particular, their overlap is part of the left hand side of the rule that is used to derive the involved transitions. In general, in the composition of two states, we might want to have a bigger overlap than strictly necessary. To illustrate this point, the processes $\overline{a}.0 \parallel P$ and $a.0 \parallel Q$, can communicate. It suffices that they communicate over the name a; however, in the parallel composition $\overline{a}.0 \parallel P \parallel a.0 \parallel Q$, the process P and Q share all free names and not only the name a. Thus, while the minimal overlap is just the channel name a, we in fact want also to share *all* common free names of P and Q in $\overline{a}.0 \parallel P \parallel a.0 \parallel Q$.

Thus, we want to extend Proposition 2.4.4 to "arbitrary" overlaps of two given states $J \to G$ and $J' \to G'$; more detailed, we start with two DPOBC-diagrams d and d' and an "admissible" overlap $G \leftarrow X \to G'$. The idea of an "admissible" is simple: the graph X should just be an extension of the minimal overlap of G and G' that is necessary for communication.

As proof technique, we want to reuse Proposition 2.4.4 by "artificially" enlarging the rule that we use to obtain the desired composition to a suitable *extended rule*.

Definition 2.4.5 (Extended rule)**.** For any rule $\rho = L \leftarrow I \to R$ and any supergraph $L \to L^+$ of L, the L^+-*instance* of ρ, denoted $\rho(L^+)$, is the lower-span of the following double-pushout diagram (if it exists).



Now, the main idea of the theorem is to extend the rule that is used in the composition just enough to obtain the "desired" overlap X as the minimal overlap of Proposition 2.4.4.

Theorem 2.4.6 (Composition of transitions)**.** *Let $s = J \to G$ and $s' = J' \to G'$ be states; moreover let $t = (J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)$ and $t' = (J' \to G') \xrightarrow{J' \to F' \leftarrow K'} (K' \to H')$ be two transition using respective partial matches D and D' into a common rule $\rho = L \leftarrow I \to R$. Let X be a common subgraph of G and G' and let $\overline{G} = G +_X G'$ be the pushout of G and G' over X.*

Now, there exists a transition $\overline{t} = (\overline{J} \to \overline{G}) \xrightarrow{\overline{J} \to \overline{F} \leftarrow \overline{K}} (\overline{K} \to \overline{H})$ that is homogeneous with respect to both t and t' (where $\overline{J}$ is the union of J and J') if there exists E such that (the black part of) the diagram in Figure 2.13 consists of three pullback squares.

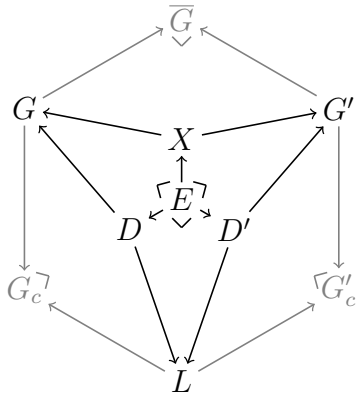


Figure 2.13: The condition for ρ-composability (in black) and some pushouts from the proof (in grey)

Proof. We start by building the pushout of $L \leftarrow E \to X$, and call $\widetilde{L}$ the pushout graph. Because EXG_cL (resp. EXG'_cL) is a composition of two pullbacks, it is itself a pullback square. Therefore, there is a unique monomorphism $\widetilde{L} \to G_c$ (resp. $\widetilde{L} \to G'_c$) making the diagram with the square commute, in other words, $\widetilde{L}$ splits $L \to G_c$ (resp. $L \to G'_c$) into two monomorphisms. By classical pushout splitting, we can split the square $EL\tilde{L}X$ into two pushouts. Since the square $EDGX$ is a pullback, there is a unique monomorphism $D^+ \to G$ such that the diagram in Figure 2.14(a) commutes. Because of pushout decomposition properties, the square $D^+\tilde{L}G_cG$ is a pushout.

A symmetric construction yields a match D'^+ for G' and $\tilde{L}$. Since $EDLD'$ is a pullback, and all vertical squares of the cube of Figure 2.14(b) are pushouts, the lower square is a pullback.

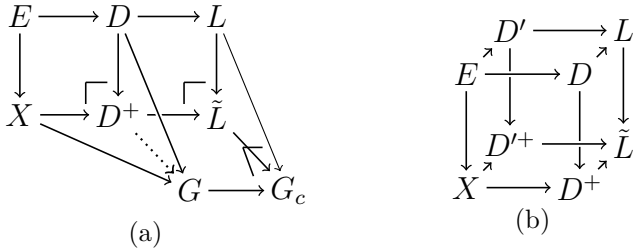


Figure 2.14

We can now construct the diagrams $d(\widetilde{L})$ and $d'(\widetilde{L})$ by the construction mentioned after Definition 2.4.5, with D^+ and D'^+ as matches. We compose them by Proposition 2.4.4 and obtain the DPOBC-diagram $\widetilde{d}$ shown in Figure 2.15(a).

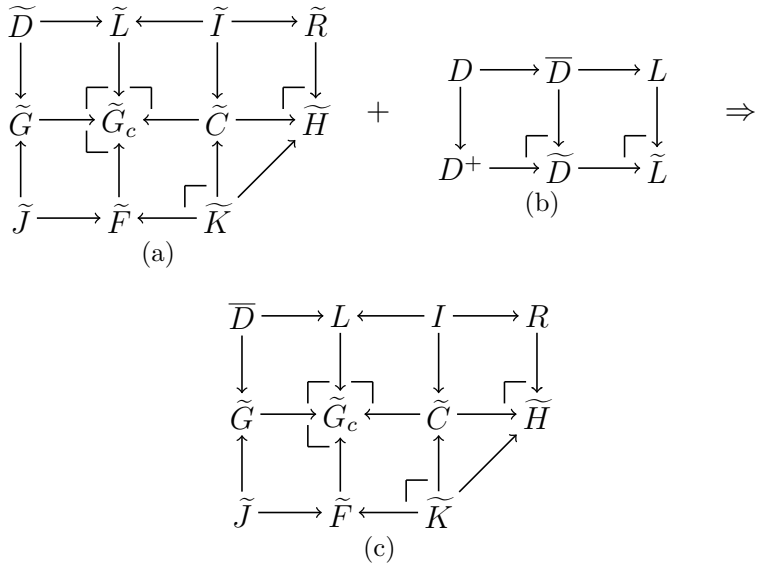


Figure 2.15: Recomposition of a DPOBC-diagram with the correct rule.

By pushout complement splitting in two of the upper-left square of the construction of $d(\widetilde{L})$, as shown in Figure 2.15(b), one obtains a match for the completion of $\widetilde{d}$ into a DPOBC-diagram with rule ρ. Thanks to the uniqueness properties of pushouts and pullbacks, it is easy to show that the same match would be constructed from d'.

It is now left to show that $\overline{G} \cong \widetilde{G}$. In fact, we not only want to prove that there exists a homogeneous transition, but also that there is one involving $\overline{G}$.

Figure 2.16(a), where all "faces" are pushout squares, shows how $\widetilde{G}$ is constructed, as well as some other parts of the composition diagram. One part of it, namely the left face of the cube, the top square and the two diamond-shaped faces are shown flattened out in Figure 2.16(b).

By uniqueness of pushouts, the outer square of Figure 2.16(b) being one, we have $\overline{G} \cong \widetilde{G}$. □

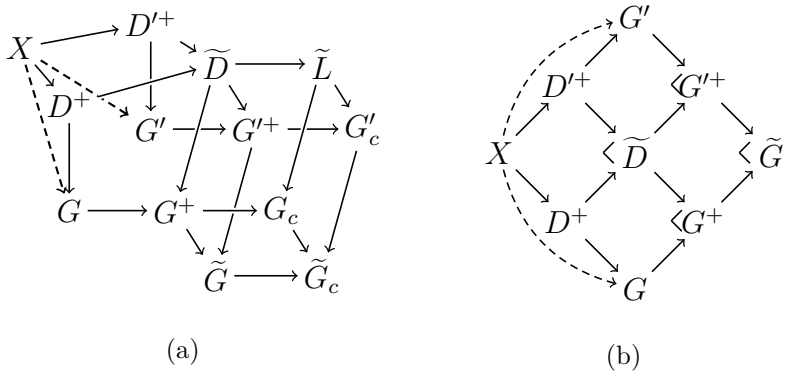


Figure 2.16

The intuition behind the condition of Theorem 2.4.6 is as follows. Whatever G and G' had in common with L, namely the partial matches D and D' respectively, are glued in $\overline{G}$ as they should be in L, i.e. by identifying the same elements, namely E. Technically, the condition ensures that the pullback of $D \to \overline{G} \leftarrow D'$ is the same as the one of $D \to L \leftarrow D'$. For instance, if D and D' contain both more than half the rule L, the gluing in $\overline{G}$ has to identify everything they have "in common" to form L, and not something "bigger". Of course, if this gluing is bigger than L (if not enough is identified), it still makes $\overline{G}$ able to evolve using this rule. But then, it is not clear how to obtain the resulting graph $\overline{H}$ from the two resulting graphs H and H', so we will not understand it as composition of transitions.

We will make use of the condition of the theorem in the next section to succinctly describe the premises of the counterpart to the communication rule of CCS.

Definition 2.4.7. With the assumptions of Theorem 2.4.6, the transitions t and t' are *ρ-composable* if there exists E such that (the black part of) the diagram in Figure 2.13 consists of three pullback squares; the transition $\bar{t}$ of the theorem is the *composition through* $\overline{G}$ of t and t' (relative to D and D').

Remark 2.4.8 (The "Meaning" of Compositionality). The word *compositionality* is used differently in different contexts. We understand compositionality as the principle that the semantics of the whole is determined by the semantics of the parts, where the (possibly composed) entities are states with interfaces. To illustrate the basic idea, take all transitions of a state as semantics; in this situation, using Theorem 2.4.6, we have that this "all transitions" semantics is compositional: each transition of the composed state can be (re-)constructed from the transitions of its components. It is an open problem how this approach to compositionality can be incorporated into the work presented in [52] where compositionality is considered on the level of rules and rewriting diagrams.

In fact, we shall only make use of the special case of Theorem 2.4.6 where X is actually a common sub-interface of the states $J \to G$ and $J' \to G'$, i.e. X is a common

subgraph of J and J'. This is in analogy to process calculi where it is crucial that parallel composition lets the composed processes only share some of their free names.

2.5 SOS semantics

We now make use of the composition result of Theorem 2.4.6 in two ways: first, as already discussed at length, we obtain a counterpart of CCS-style communication for graph rewriting; second, we can dispense with a number of "superfluous axioms" for basic actions.

Looking at CCS again, we can notice a difference with 3L in the definition of axioms. In fact, in 3L, the application of the rewriting rules themselves appear as basic actions, which is not the case in CCS where we cannot have both actions a and $\overline{a}$ in a single label. The communication rule covers the case where both are performed complementary, which corresponds exactly to the application of a reduction rule. Conversely, in 3L, any match for a left-hand side of a rule yields a basic action, in particular the empty graph. Now that we have an equivalent of the CCS-composition, we can remove from the set of basic actions certain "superfluous" ones, which can then be reconstructed by addition of a composition rule.

As "good" matches for a rule in graphs we shall take the irreducible elements in the lattice of subgraphs of the left hand side of the rule. Using irreducible graphs, we define *atomic* actions, which use irreducible graphs as partial matches. However, we shall show that we can obtain all basic actions by applying the composition rule to the atomic ones. This allows us to define a system that is exactly as expressive as 3L (and hence DPOBC) with one extra rule, but a smaller number of axioms.

Definition 2.5.1 (Irreducible graph)**.** Let G be a hypergraph. A *(non-trivial) decomposition* of G is a pair of inclusions $A \to G \leftarrow B$ such that G is the union of A and B and $G \neq A$ or $G \neq B$. A hypergraph G is *irreducible* if it has no non-trivial decomposition.

Thus, an irreducible graph cannot be decomposed into strictly smaller graphs. Clearly, a single node is an irreducible graph, but a graph composed of two single nodes is not. This formal definition fits the intuition of "atomic" hypergraphs.

Fact 2.5.2. *The only irreducible hypergraphs are the single vertex graph and all graphs that consist of a single hyperedge that is incident to every node.*

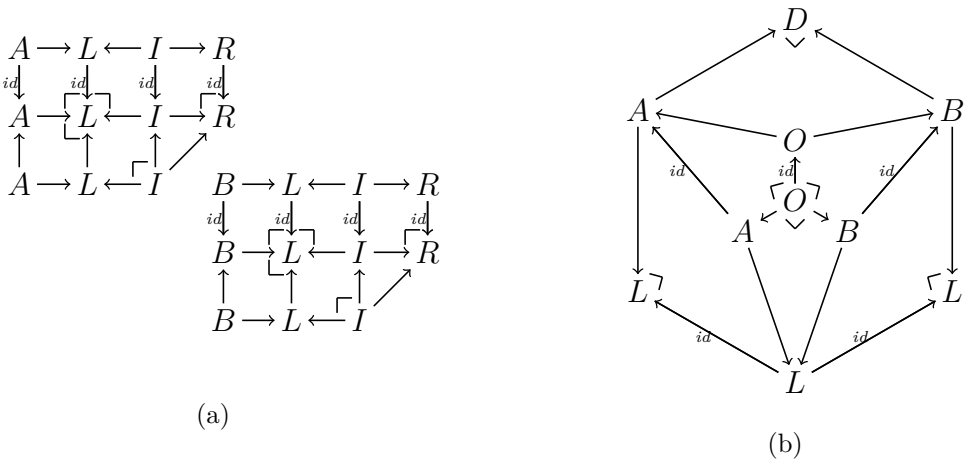
We now have all concepts to inductively define a transition system for graph rewriting in analogy to CCS. The system is summarized in Table 2.2. We call the system SOSBC, since it is an SOS-like definition for borrowed contexts (as we shall show below).

Formally, the first rule describes a family of *Atomic Actions*, which are the basic actions in which the partial match is an irreducible graph. The second and third rule are taken from 3L and have already been discussed in Section 2.3. Finally, the last rule is the one justified by the composition theorem, i.e. Theorem 2.4.6.

We now show that this system is exactly as expressive as 3L. We have already seen that it is included in it. Indeed, every transition derivable in SOSBC is also in 3L (trivially for the first three rules, and by Section 2.4 for the last one). It then suffices to show that every transition derivable in 3L is also in SOSBC. It is trivially true for interface narrowing and compatible contextualization. It is left to show that every basic action can be derived in SOSBC.

Lemma 2.5.3 (Basic action decomposition)**.** *Let $\rho = L \leftarrow I \rightarrow R$ be a rule and D a partial match for L. Let $A \rightarrow D \leftarrow B$ be a decomposition of D. Then $A \rightarrow A$ and $B \rightarrow B$ are ρ-composable through D and the result of the composition is exactly the axiom transition $(D \rightarrow D) \xrightarrow{D \rightarrow L \leftarrow I} (I \rightarrow R)$.*

Proof. We first show that the condition of Theorem 2.4.6 holds. As $A \rightarrow D \leftarrow B$ is a decomposition of D, the morphisms D is the union of A and B. Thus, let $A \leftarrow O \rightarrow B$ be the pullback of $A \rightarrow D \leftarrow B$ to obtain a pushout square. Since D is a partial match for L, then so are A and B, and therefore their corresponding basic action transitions exist. They are justified by the DPOBC-diagrams in Figure 2.17(a), which lead to the construction of the diagram 2.17(b).



(a) (b)

Figure 2.17

After applying the construction of Theorem 2.4.6, it is easy to check that the resulting composition of the two transitions is the transition $(D \rightarrow D) \xrightarrow{D \rightarrow L \leftarrow I} (I \rightarrow R)$.

□

Theorem 2.5.4 (Basic Completeness)**.** *Any basic action of the* 3L*-system can be obtained from atomic actions and the composition rule of the* SOSBC*-system.*

Proof. Let $t = (D \rightarrow D) \xrightarrow{D \rightarrow L \leftarrow I} (I \rightarrow R)$ be a basic action and l be the size of D, i.e. the number of hyperedges. By applying the decomposition lemma (Lemma 2.5.3)

- **Atomic Actions**

$$\frac{}{(D \to D) \xrightarrow{D \to L \leftarrow I} (I \to R)} \quad \text{where} \quad \begin{array}{l}(L \leftarrow I \to R) \in \mathcal{S} \\ \text{and } D \text{ irreducible} \\ \text{with } D \to L\end{array}$$

- **Interface Narrowing**

$$\frac{(J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)}{(J' \to G) \xrightarrow{J' \to F' \leftarrow K'} (K' \to H)} \quad \text{where} \quad \begin{array}{l}C = J \to J \leftarrow J' \\ \text{and } J' \to F' \leftarrow K' = C[J \to F \leftarrow K]\end{array}$$

- **Compatible Contextualization**

$$\frac{(J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)}{C[J \to G] \xrightarrow{C[J \to F \leftarrow K]} \overline{C}[K \to H]} \quad \text{where} \quad \begin{array}{l}C = J \to E \leftarrow \overline{J} \text{ compatible with } J \to F \leftarrow K \\ \text{and } \overline{C} = (J \to F \leftarrow K)[C]\end{array}$$

- **Composition**

$$\frac{t = (J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H) \qquad t' = (J' \to G') \xrightarrow{J' \to F' \leftarrow K'} (K' \to H') \qquad J \leftarrow X \to J'}{\bar{t} = ((J +_X J') \to (G +_X G')) \xrightarrow{(J +_X J') \to \overline{F} \leftarrow \overline{K}} (\overline{K} \to \overline{H})}$$

$$\text{where} \quad \begin{array}{l}t \text{ and } t' \text{are } \rho\text{-composable} \\ \text{and } \bar{t} \text{ is their composition through } G +_X G'\end{array}$$

Table 2.2: Axioms and rules of the SOSBC-system.

repeatedly until all the subgraphs of D are decomposed into irreducible graphs, one obtains a binary tree whose root is t and nodes are basic actions. Since l is finite, and the decomposition lemma decreases the size of graphs, this process is finite, and the leaves of the tree are atomic actions. □

Theorem 2.5.5 (Soundness and completeness)**.** *Let $\mathcal{S}$ be a graph transformation system. Then there is a* BC*-transition*

$$(J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)$$

if and only if it is derivable in the SOSBC*-system.*

Proof. Atomic actions, interface narrowing and contextualization are part of the 3L-system, therefore they are derivable as BC-transitions. By construction, the composition yields derivable transitions in BC too. Conversely, every BC-transition is derivable in 3L, and by Theorem 2.5.4, is derivable in SOSBC. □

In fact, every derivation tree obtained by the process that is described in the theorem will have the same structure: a certain number n of Atomic Actions, followed by $n - 1$ applications of the composition rule. Then a Compatible Contextualization

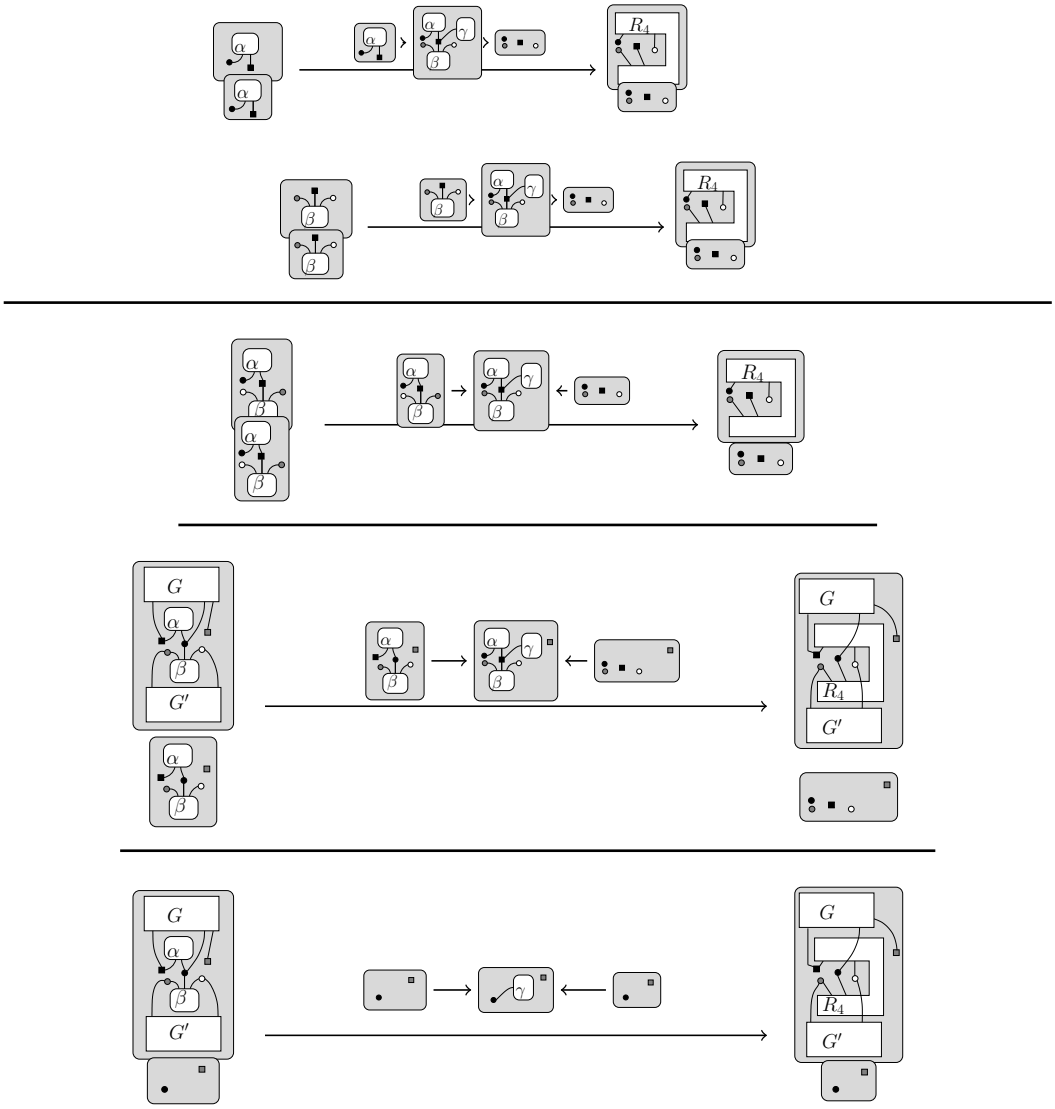


Figure 2.18: An example of a derivation in the SOSBC-system.

and an Interface Narrowing, as in the case of the 3L system. Consider the following example of a derivation tree.

Example 2.5.6 (Composition of transitions). As example, consider the derivation tree of SOSBC for the final transition of Example 2.4.1, which is shown in Figure 2.18.

2.6 Application to interaction nets

Process calculi, such as CCS and the π-calculus, have a so-called *communication rule* that allows to synchronize sub-processes to perform silent actions. The involved process terms have complementary actions that allow to interact by a "hand-shake".

As we have seen before, it is not the case in general graph rewriting theory. In interaction nets, we get close to this idea, since all interactions are binary. The difference with most common process algebras is that there are several possible interactions.

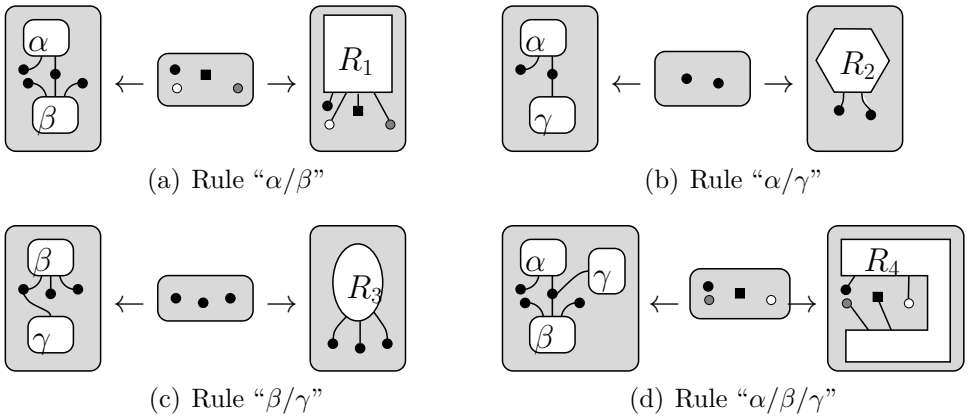
To elaborate on this using the metaphor of handshakes, assume that we have an agent that needs a hand to perform a handshake or to deliver an object. If we observe this agent reaching out for another hand, we cannot conclude from it which of the two possible actions will follow. In general, even after the action is performed, it still is not possible to know the decision of the agent – without extra information, which might however not be observable. However, with suitable assumptions about the "allowed actions", all necessary information might be available.

In this section, we will address binary rules from a global point of view. Instead of stating that we allow only this kind of rules, we will split all rules in two, considering half-left-hand sides as generalized agents. This is more general than interaction net rules, since agents will be able to somehow overlap each other. As an example, consider a ternary rule for cells α, β, γ given some connections between them. We split this rule in three rules, one for the pair $\alpha \bowtie (\beta, \gamma)$, one for $\beta \bowtie (\alpha, \gamma)$ and one for $\gamma \bowtie (\alpha, \beta)$.

We then give a few conditions under which it is possible to retrieve the reduction rule from the label and the original state. The first of these conditions enlightens us on the reason why the composition rule is so neat in usual name-based process calculi, while the second one leads directly to simply wired interaction nets.

The running example that will be used to illustrate this section is the following.

Example 2.6.1. The system $\mathcal{S}_{ex} = (\Lambda, \mathcal{R})$ will be the following one in the sequel: $\Lambda = \{\alpha, \beta, \gamma, \dots\}$ such that $\mathtt{ar}(\alpha) = 2$, $\mathtt{ar}(\beta) = 3$ and $\mathtt{ar}(\gamma) = 1$; moreover $\mathcal{R}$ is the set of rules given in Figure 2.19 where the R_i are different graphs, e.g. hyperedges with suitable arity.



(a) Rule "α/β" (b) Rule "α/γ"

(c) Rule "β/γ" (d) Rule "$\alpha/\beta/\gamma$"

Figure 2.19: Reaction rules of $\mathcal{S}_{ex}$.

First, we recall that DPOBC-diagrams can be composed under certain circumstances, as detailed in Proposition 2.4.4.

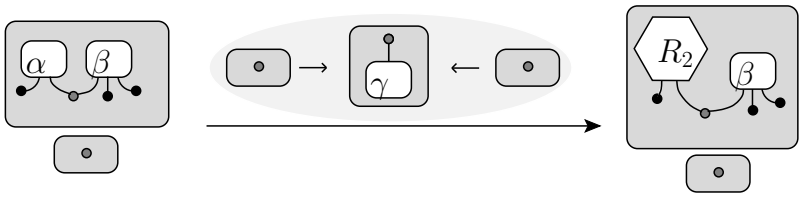
Fact 2.6.2. *Let*

$$(J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H) \quad \text{and} \quad (J' \to G') \xrightarrow{J' \to F' \leftarrow K'} (K' \to H')$$

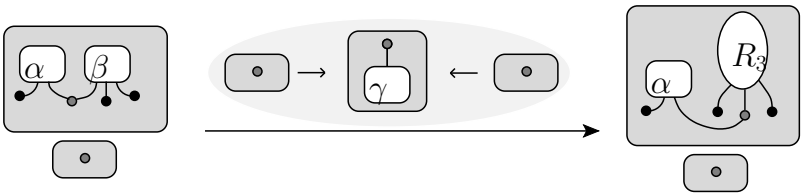
be two transitions obtained from the same rule $\rho = L \leftarrow I \to R$ *by* DPOBC*-diagrams* $\mathtt{D}_1$ *and* $\mathtt{D}_2$*. Then, it is possible to build a* DPOBC*-diagram with same rule for the composition of* $J \to G$ *and* $J' \to G'$ *along some common interface* $J \leftarrow J^L_D \to J'$*.*

However, we emphasize at this point, that derivability of a counterpart of the communication rule of CCS is not the same question as the composition of pairs of transitions that come equipped with *complete* BC-diagrams. To clarify the problem, consider the following example where we cannot infer the used rule from the transition label.

Example 2.6.3. Let G be a graph composed of two edges α and β and consider a transition label where an edge γ is "added". Then it is justified by both rules α/γ and β/γ (see Figure 2.20).



(a) A transition from rule α/γ



(b) A transition from rule β/γ

Figure 2.20: Same transition label for different rules.

Our purpose is exactly to avoid this problem by restricting to suitable classes of graph transformation systems. For this, we shall focus on the derivation of "silent" transitions in the spirit of the communication rule of CCS.

Definition 2.6.4 (Silent label)**.** A label $J \to F \leftarrow K$ is *silent* or τ if $J = F = K$; a *silent transition* is a transition with a silent label.

Intuitively, a silent transition is one that does not induce any "material" change that is visible to an external observer that only has access to the interface of the states. Hence, in particular, a silent transition does not involve additions of the environment during the transition. Moreover, the interface remains unchanged. This latter requirement does not have any counterpart in process calculi, as the interface is given

implicitly by the set of all free names. (In graphical encodings of process terms [7] it is possible to have free names in the interface even though there is no corresponding input or output prefix in the term.)

Now, with this special case in mind, for a given rule $L \leftarrow I \rightarrow R$ we can illustrate the idea of complementary actions as follows. If a graph G contains a subgraph D of L and moreover a graph G' has the complementary subgraph of D in L in it, then G and G' can be combined in a big graph $\overline{G}$ that has the whole left hand side L as a subgraph and thus $\overline{G}$ can perform the reaction. A natural example for this are interaction nets of course. The intuitive idea of complementary (basic) actions is captured by the notion of *active pairs*.

Definition 2.6.5 (Active pairs)**.** For any inclusion $D \rightarrow L$, where $D \neq L$ and for all nodes v of D, $\deg(v) > 0$, let the following square be its initial pushout

$$\begin{array}{ccc} J_D^L & \longrightarrow & \widehat{D}^L \\ \downarrow & & \downarrow \\ D & \longrightarrow & L \end{array},$$

i.e. $\widehat{D}^L$ is the smallest subgraph of L that allows for completion to a pushout. We call $\widehat{D}^L$ the *complement of D in L* and J_D^L the *minimal interface of D in L* and we write $\{D, D'\} \equiv L$ if $D' = \widehat{D}^L$. The set of *active pairs* is

$$\mathbb{D} = \Big\{ \{D, \widehat{D}^L\} \mid \forall L \leftarrow I \rightarrow R \in \mathcal{R} \Big\}.$$

Abusing notation, we also denote by $\mathbb{D}$ the union of $\mathbb{D}$.

It is easy to verify that the complement of $\widehat{D}^L$ in L is D itself and that its minimal interface is also J_D^L. It is the set of "acceptable" partial matches in the sense that they do not yield a τ-reaction on their own. Indeed, if D is equal to L, then the resulting transition of this partial match is a τ-transition. And if it is just composed of vertices, its complement is L and thus not acceptable.

Example 2.6.6 (Active pairs). In our running example, the set $\mathbb{D}$ of our example is

$$\Big\{\{\alpha, \beta\}, \{\alpha, \gamma\}, \{\beta, \gamma\}, \{\alpha, \beta + \gamma\}, \{\alpha + \beta, \gamma\}, \{\alpha + \gamma, \beta\}\Big\}.$$

The minimal interface of any pair is a single vertex.

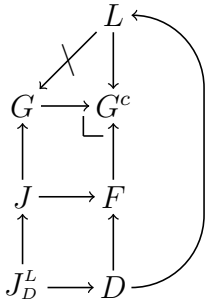
This completes the introduction of preliminary concepts to tackle the issues that have to be resolved to obtain "proper" compositionality of transitions. We can clearly see how this is usable for interaction nets, where redexes are naturally defined as active pairs in this sense.

2.6.1 Towards a partial solution

We shall address the problem of finding the right reaction rule in some more detail. We start by considering the left half of labels, which intuitively describe possible borrowing

actions from the context. Relative to this, we define the *admissible rules* as those rules that can be used to let states evolve while borrowing the specified "extra material" from the environment.

Definition 2.6.7 (Admissible rule)**.** Let $J \to G$ be a state and let $J \to K$ be an inclusion (which represents a possible contribution of the context). A rule ρ is *admissible (for $J \to K$)* if $L \not\to G$ and it is possible to find $D \in \mathbb{D}$ and L the left-hand side of ρ, such that the following diagram commutes



where $J^L_D \to D$ is the minimal interface of D in L. We call D the *rule addition*.

This just means that G *can* evolve using the rule ρ if D is added at the proper location.

Proposition 2.6.8 (Pre-compositionality)**.** *Let*

$$(J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H) \quad \text{and} \quad (J' \to G') \xrightarrow{J' \to F' \leftarrow K'} (K' \to H')$$

be two transitions such that a single rule ρ is admissible for both, and let D and D' be their respective rule additions. If $\{D, D'\} \in \mathbb{D}$, it is possible to compose G and G' into a graph $\overline{G}$ in a way to be able to derive a τ-transition using rule ρ.

Proof. We first show that in such a case, $D' \to G$ and the pushout of $G \leftarrow D' \to L$ is exactly G^c. Similarly, $D \to G'$ and the pushout of $G' \leftarrow D \to L$ is exactly G'^c. Then, it is easy to see that it is possible to build the DPOBC-diagram $\mathtt{D}_1$ using rule ρ on G (respectively G') yielding the transition $(J \to G) \xrightarrow{J \to F \leftarrow K_1} (K_1 \to H_1)$ for some K_1, H_1 (respectively $\mathtt{D}_2$ yielding the transition $(J \to G) \xrightarrow{J \to F \leftarrow K_2} (K_2 \to H_2)$ for some K_2, H_2), and then compose $\mathtt{D}_1$ and $\mathtt{D}_2$ using Proposition 2.4.4.

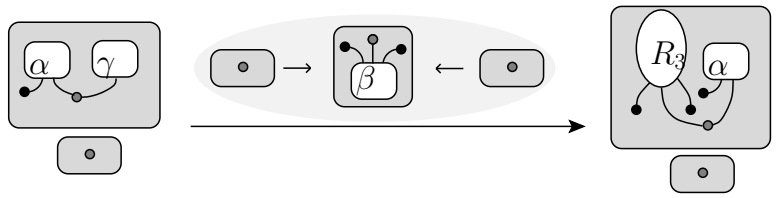
This follows from $\{D, D'\} \in \mathbb{D}$ and $\overline{G} \equiv \overline{G^c}$. Indeed, $\overline{E} = L$ so the top left morphism of the composed DPOBC-diagram is an isomorphism and so are the ones under it, using basic pushout properties. □

This first result motivates the following definition.

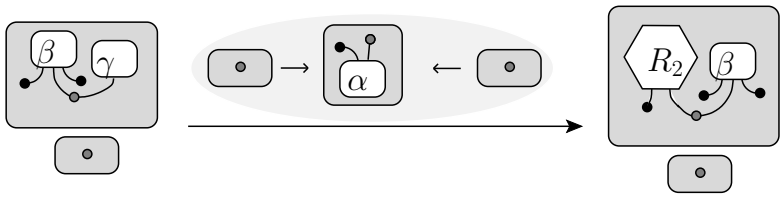
Definition 2.6.9 (τ-compatible)**.** In the situation of Proposition 2.6.8, we say the two transitions are *τ-compatible*.

Remark 2.6.10. In general, the result of the τ-transition cannot be obtained from H and H'. So we do not yet talk about compositionality.

Example 2.6.11. Let G be a graph composed of two edges α and γ and G' of two edges β and γ (see Figure 2.21). Then the rule α/β is admissible for both transitions and moreover they are τ-compatible. The rule α/β yields the respective rule additions. "Glueing" G and G' by their interface results in a graph with edges α, β and two γs; the latter graph can perform a τ-reaction from rule α/β, which however does not give the desired result since the target state is not the "expected composition" of H and H'. In other words, although we have been able to construct a τ-transition, it is not the composition of the original transitions.



(a) A transition from rule β/γ



(b) A transition from rule α/γ

Figure 2.21: τ-compatible, but not composable: different rules.

We can see from the examples here that the difficulty of defining a composition of transitions comes mainly from three facts. The first is that a partial match can have several subgraphs triggering a reaction. This is dealt with by the construction of the set of active pairs. The second is the possibility to connect multiple edges together, not knowing which one exactly is consumed in the reaction. Finally, a given edge can have multiples ways of triggering a reaction.

2.6.2 Sufficient conditions

We now give two frameworks in which neither of the two last problems do occur. Avoiding each of them separately is enough to define compositionality properly. Both cases are inspired by the study of interaction net systems, which can be represented in the "obvious" manner as graph transformation systems. We ironize on the evidence of such a representation as it is not at all straightforward. Such a representation is detailed for instance by Banach in [3].

In systems embodying one of the two conditions we give, the DPOBC-diagram built from an admissible rule of a transition is necessarily the one that has to be used to derive the transition. In one case, it works for essentially the same reasons as in CCS:

every active element can only interact with a unique other element, such as a vs. $\overline{a}$, b vs. $\overline{b}$. In the other one, the label itself is not enough, but since we also know where it "connects" to the graph, it is possible to "find" the partner that was involved in the transition.

We introduce interaction graph systems, which are the counterpart of interaction nets in the hypergraph rewriting setting and vocabulary introduced so far. We fix a labeling alphabet Λ.

Definition 2.6.12. An *activated pair* is a hypergraph L on Λ composed of two hyperedges e and f and a node v such that v appears exactly once in $\mathrm{cnct}(e)$ and once in $\mathrm{cnct}(f)$. If v is the i-th incident vertex of e labeled α and the j-th incident vertex of f labeled β, we denote the activated pair by $e_i \bowtie f_j$ and label it by $\alpha_i \bowtie \beta_j$.

An *interaction graph system* $(\Lambda, \mathcal{R})$ is given by a set of reaction rules $\mathcal{R}$ over hypergraphs on Λ where all left-hand side of rules are activated pairs, and nodes are never deleted, i.e. for any rule $\rho = L \leftarrow I \rightarrow R$,

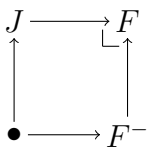
- L is an activated pair;
- for any node v, $v \in L \Rightarrow v \in I$.

Note that for any interaction graph system, the set $\mathbb{D}$ is composed of pairs $\{D, D'\}$ where each of them is composed of an edge and its connected vertices. Also the minimal interface of any active pair $\{D, D'\}$ is a single node. It is also the case that it is enough for interfaces to be composed of vertices only.

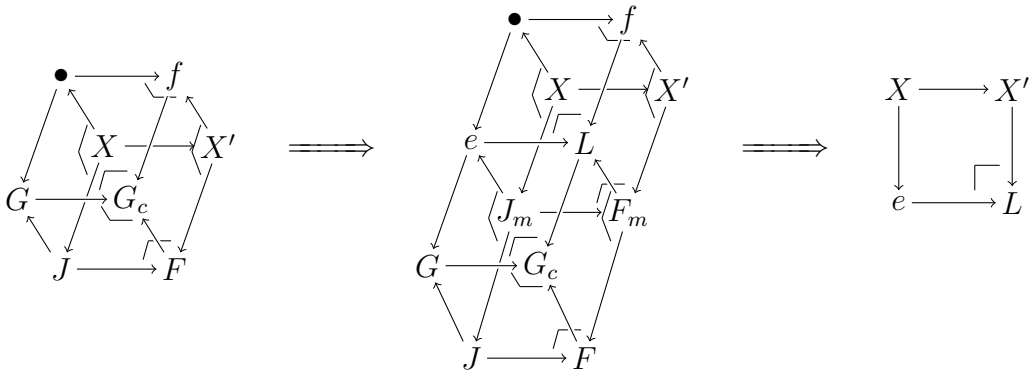
We first take advantage of the form of LHS in interaction graph systems to give a nice condition of τ-compositionality. We will use the fact that an activated pair $e_i \bowtie f_j$ has exactly one node v that appears exactly once in $\mathrm{cnct}(e)$ and once in $\mathrm{cnct}(f)$. In the following we denote by $\bullet$ the unique interaction graph composed of a single node. In diagramatical terms, it means that

since an activated pair is always connected.

Lemma 2.6.13. *Let $\mathcal{S}$ be an interaction graph system and $(J \rightarrow G) \xrightarrow{J \rightarrow F \leftarrow K} (K \rightarrow H)$ be a reduction of state $J \rightarrow G$ in that system. Then there exist a unique injection $\bullet \rightarrow J$ such that $\bullet \rightarrow J \rightarrow F$ has a pushout complement $\bullet \rightarrow F^- \rightarrow F$:*



Proof. Let us show first that such an injection exists. We consider the DPOBC-diagram d justifying the reduction, using rule for $L = e \bowtie f$. From its first column, we build the pullback $J \leftarrow X \rightarrow \bullet$ of $J \rightarrow G \leftarrow \bullet$ and the pullback $F \leftarrow X' \rightarrow f$ of $F \rightarrow G_c \leftarrow f$, yielding an already famous cube with four pullbacks, of which two are pushouts, implying the two "unknown" ones to be pushouts also. By pullback splitting, we split the cube into two cubes with all corresponding pushouts and pullbacks, which implies that $X' \leftarrow X \rightarrow e$ has a pushout $e \rightarrow L \leftarrow X'$:



By the previous remarks, X cannot be the empty graph, so $X = \bullet$ (and $X' \equiv f$), so $X \rightarrow J$ is our desired $\bullet \rightarrow J$.

Why is it unique? Well, imagine two pushout squares

Since all morphisms are injection, the squares are also pullbacks, and so there are injections i and j from $\bullet \rightarrow \bullet$ s.t. $f \circ j = f'$ and $f' \circ i = f$. But there is only one possible injection $\mathtt{id} = \bullet \rightarrow \bullet$, so $f = f'$ and $g = g'$.

□

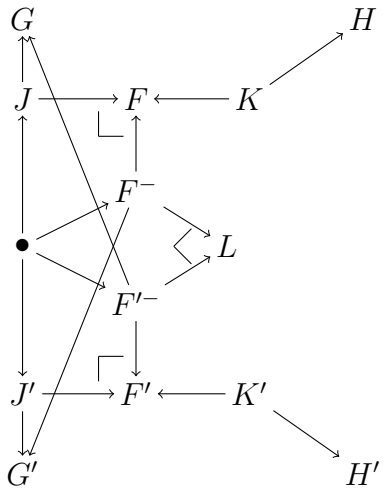
We have shown that, given any reduction label for a state $J \rightarrow G$, their exists a unique non-interfering narrowing context $\bullet \rightarrow J \leftarrow J$. Its corresponding narrowing label is in fact the most interesting kind of labels in interaction nets. It represents some kind of minimality for labels. It would make sense for a theory of structural semantics for interaction nets like ours to replace atomic actions on "full states" (see Table 2.2) by atomic actions on states of the form $\bullet \rightarrow D$. We could then probably omit the interface narrowing rule. We will consider such a structure in future work.

We now give a new sufficient condition for labels to be composable. Its advantage is that it does not involve to imagine which rule is admissible for the states but it actually gives it.

Theorem 2.6.14 (τ-compatibility)**.** *Let*

$$(J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H) \quad \textit{and} \quad (J' \to G') \xrightarrow{J' \to F' \leftarrow K'} (K' \to H')$$

be two reductions. If there exist morphisms $F^- \to G'$ and $F'^- \to G$ such that the following diagram commutes, and $F^- \bowtie F'^-$ is the LHS of a rule ρ, then $J \to G$ and $J \to G'$ are τ-compatible.



Proof. Show that ρ is admissible for $J \to G$ and $J' \to G'$. □

Remark 2.6.10 is of course still valid. What we know from the condition is that it is possible to compose states $J \to G$ and $J' \to G'$ in a state $\overline{J} \to \overline{G}$ s.t. this latter has a silent transition. We even know which rule yields that transition. It is still not the case that the result of the silent transition is linked in any way to H and H'. In our example shown in Figure 2.21, the labels are already the "narrowest" possible and they fulfill the conditions of the theorem. We are therefore sure it is possible to combine two nets, and one can see it. The result is not in any way the composition of the right-hand side nets.

It is true, on the other hand, that the theorem hold in the general case of graph rewriting. If it is possible to build the diagram above from the two label, then the states are τ-compatible. But this means to find a common interface for which both labels have a narrowing interface involving half of an activated pair and such that the whole thing commutes. It is in general a rather difficult task. In interaction graphs, it is quite natural, because of the unicity of the narrowing.

We finally give two frameworks for which τ-compatibility implies necessarily compositionality.

Hypergraphs with unique partners In multiwire interaction nets, we lose the unicity of the rule for a given transition label. It can be recovered by another condition.

Definition 2.6.15 (Unique partners)**.** Let $\mathbb{I} = (\Lambda, \mathcal{R})$ be an interaction graph system. We say it is *with unique partners* if for any $\alpha \in \Lambda$, $\forall i \leq \mathtt{ar}(\alpha)$, exists a unique $\beta \in \Lambda$ and a unique $j \leq \mathtt{ar}(\beta)$ such that $\alpha_i \bowtie \beta_j$ is the label of a left-hand side of a rule in $\mathcal{R}$.

Lemma 2.6.16. *Let $J \to G$ a state of $\mathbb{I}$ and $(J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)$ a non-τ reaction label. Then there is exactly one admissible rule ρ for this transition.*

Simply wired hypergraphs On the other hand, simple wires seem sufficient to gain the power of finding the used rule form the label of the action. We define the equivalent of simple wires for hypergraphs.

Definition 2.6.17. Let $N = (E, V, \ell, \mathrm{cnct})$ be a hypergraph on Λ.

The graph N is *simply wired* if $\forall v \in V$, $\deg(v) \leq 2$. When $\deg(v) = 1$, we say that v is *free*.

In other words, vertices are only incident to at most two edges of a graph. First, we have to notice that it is possible to define the category of simply wired hypergraphs as a sub-category of hypergraphs.

The purpose of the interface of a graph being observability in the sense of the possibility to connect some context to it, in simply wired hypergraphs, it is meaningless for a vertex that is already connected to two edges to be in the interface.

Definition 2.6.18 (Lafont interaction graph system)**.** A *Lafont interaction graph* is a simply connected graph such that its interface consists of free vertices only. A *Lafont system* $\mathbb{L} = (\Lambda, \mathcal{R})$ is given by reaction rules over Lafont interaction graphs.

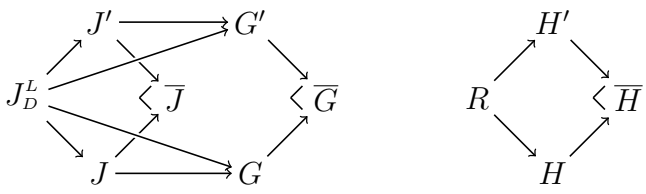
Lemma 2.6.19. *Let $J \to G$ be a state of $\mathbb{L}$ and $(J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)$ a non-τ transition. Then there is exactly one admissible rule ρ for this transition.*

Finally, we conclude our investigation with the following positive result.

Theorem 2.6.20 (Compositionality)**.** *Let $\mathcal{S} = (\Lambda, \mathcal{R})$ be a Lafont interaction graph system, or an interaction graph system with unique partners. Let $\mathbb{D}$ be its set of active pairs.*

Let $t_1 = (J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)$ and $t_2 = (J' \to G') \xrightarrow{J' \to F' \leftarrow K'} (K' \to H')$ be two transitions in $\mathcal{T}$ and D and D' their respective rule additions.

If $\{D, D'\} \equiv L \in \mathbb{D}$, let $\overline{G}$ and $\overline{H}$ are described by the following diagrams



where $J_D^L \to J$ and $J_D^L \to J'$ are the inclusions from the admissibility of ρ for states $J \to G$ and $J' \to G'$ (Definition 2.6.7).

Then

$$(\overline{J} \to \overline{G}) \xrightarrow{\overline{J} \to \overline{J} \leftarrow \overline{J}} (\overline{J} \to \overline{H}).$$

Sketch of proof. By Lemma 2.6.19 or 2.6.16, there exists exactly one rule $\rho \in \mathcal{R}$ with L as a left-hand side that allows to derive transitions t_1 and t_2 – it is indeed the same rule for both. Let $\mathtt{D}$ be the composition diagram of the DPOBC-diagrams justifying the transitions (see Proposition 2.4.4).

It is first shown that $\overline{G} \equiv \overline{G}_c$. Since the upper and lower left squares of $\mathtt{D}$ are pushouts we can infer that $\overline{D} \equiv L$ and $\overline{J} \equiv \overline{F}$. Finally, since no vertex is deleted (see Definition 2.6.12), we have $\overline{J} \to \overline{C}$ and thus $\overline{K} \equiv \overline{J}$.

So $\mathtt{D}$ is a BC-diagram of a τ-reaction from $\overline{J} \to \overline{G}$ to $\overline{J} \to \overline{H}$. □

We haven't achieved structurality, since the resulting reduction label depends highly on the full recomposition of both DPOBC-diagram, especially in order to build $\overline{H}$. Even worse, in some sense, the capability of interaction depends on the knowledge of the way the LHS of a rule is to be found in each state, in some sense, reconstructing the diagrams already.

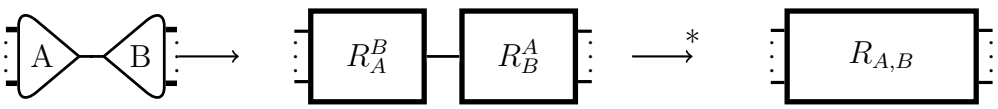
2.6.3 Particular means for particular ends

We can see how difficult it is to define a real communication rule for general graph rewriting. In this section we take advantage of the special form of LHSs in interaction nets, and cheat a little about the RHSs in order to achieve our purpose. We take inspiration in SOSBC, but define them on something closer to action then to reduction. By action, we mean half-a-reduction: an agent replaced by half-a-RHS, as we did in the intuitive semantics in the first section of this chapter. Remember that the purpose in defining a communication rule is to get a structural semantics.

Anyhow, as we can see, even though the construction of the combinations of G and G' can be rather simple, combining H and H' is only possible if one has knowledge of R as RHS of the involved rule. This goes against the structurality: one should hope to define the reduction rule from the pieces, in our case, from the halves.

We discuss the issue for simply-wired interaction net systems $\mathcal{S}$ which have cells δ, γ and the $\gamma \bowtie \gamma$ and $\delta \bowtie \delta$ rules of Lafont combinators. We suppose also that the system has the splitting property of 1.1.5. We will actually prove this property in the last chapter in the case of interaction nets with simple wires, so we only restrict to this case.

Let $A \bowtie B = L \leftarrow I \rightarrow R = A \triangleleft \triangleright B$ a rule of $\mathcal{S}$. Let R be split into R_A^B and R_B^A, joined by a unique wire between the roots of σ_A^B from R_A^B and σ_B^A from R_B^A:



We introduce new rewriting rules, which are not correct as interaction net rewriting rule and only serve as a technicality to define actions. We consider in the following

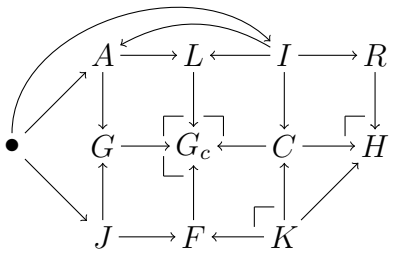
that for any rule $A \bowtie B$, I_A^B is the complete set of ports of cell A:

$$\rho_A^B := \quad L \leftarrow I_A^B \rightarrow R_A^B.$$

We thus consider the principal port of A involved in rule L to be sent to the root of σ_A^B in R_A^B, and have an inclusion $I_A^B \rightarrow A$.

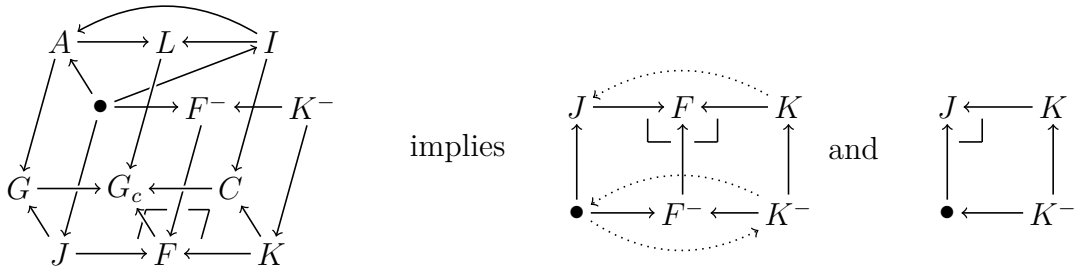
In the following, we drop the fastidious sub- and superscripts of the elements of the new kind of rules. Unless specified otherwise, I, R, C, K, H are the ones for the corresponding half-rule.

Let $J \rightarrow G$ be an interaction net containing A with free principal port p (*i.e.* $p \in J$). All the conditions are summarized by the following commutative diagram:



Proposition 2.6.21 (Special label form)**.** *In an interaction graph system, rules of the form ρ_A^B yield labels of the form $J \rightarrow F \leftarrow J$, where both injections are in fact the same.*

Proof. We construct the narrowing label from Theorem 2.6.14, which gives two pushout squares:



Indeed, the two top squares and the existence of an injection $I \rightarrow A$ implies the existence of an injection $C \rightarrow G$. By usual pullback combinations and splitting, this implies the existence of an injection $K \rightarrow J$ and $K^- \rightarrow \bullet$ (with everything commuting and the square above on the right is a pushout).

On the other hand, $\bullet \rightarrow I$ implies $\bullet \rightarrow C$. Since $\bullet \rightarrow F$, then there is a unique injection $\bullet \rightarrow K$ with everything commuting. Finally, since the lower right square is a pushout of injections, it is a pullback, so the is an injection $\bullet \rightarrow K^-$, keeping the diagram commutative.

Therefore, $K^- = \bullet$, and since there is only one possible $\bullet \rightarrow \bullet$ injection, $K^- \rightarrow \bullet = id$, and so does $K \rightarrow J$. □

In this proof, we did not use the knowledge of which node is image to the principal port of A in R. If we say demand it to be the root of the tree σ_A^B, we get reductions $(J \to G) \xrightarrow{J \to F \leftarrow J} (J \to H)$ where $J \to H$ sends the involved node into the root of σ_A^B in H.

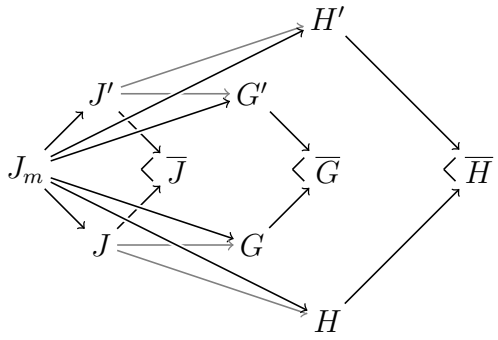
If we consider systems with unique partners or simply wired, we know that τ-compatibility implies compositionality. This allows for the particular form of the compositionality theorem:

Theorem 2.6.22 (Compositionality)**.** *Let $(\Lambda, \mathcal{R})$ be a Lafont interaction graph system, or an interaction graph system with unique partners. Let $\mathbb{D}$ be its set of active pairs.*

Consider a new system $\mathcal{T} = (\Lambda, \mathcal{R}')$ in which, for each $A \bowtie B \in \mathbb{D}$ the corresponding rules ρ_A^B and ρ_B^A to be in $\mathcal{R}'$.

Let $t_1 = (J \to G) \xrightarrow{J \to F \leftarrow K} (K \to H)$ and $t_2 = (J' \to G') \xrightarrow{J' \to F' \leftarrow K'} (K' \to H')$ be two non-τ transitions and D and D' their respective rule additions.

If $\{D, D'\} \equiv L \in \mathbb{D}$, let $\overline{G}$ and $\overline{H}$ are described by the following diagrams



where J_m is any subgraph of both J and J' identifying $\bullet$ from J and J'. Then we have the following labeled transition in $\mathcal{S}$:

$$(\overline{J} \to \overline{G}) \xrightarrow{\overline{J} \to \overline{J} \leftarrow \overline{J}} (\overline{J} \to \overline{H}).$$

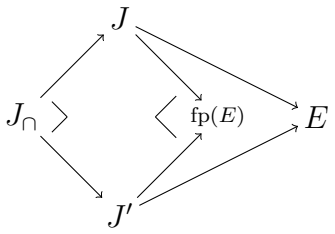
Proof. We made sure by forcing the image in the corresponding R of the principal port of each rule addition that the pushout of $H \leftarrow R \to H'$ is the same as the pushout of $H \leftarrow J_m \to H'$. □

Another important feature of interaction nets in comparison with general graph rewriting is the status of the interface. In interaction nets, the interface is the set of free ports. We cannot therefore use interface narrowing and compatible contextualization as we please.

Interface narrowing is admissible in interaction graphs if we consider we can "plug" something into a free port and make it bound, something that translates the CCS new name operator. It is usually not the case anyhow in interaction nets.

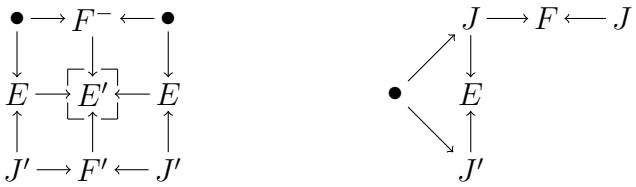
Contextualization on the other hand can bound some free ports, but also add some new free ports. So we cannot restrict ourselves to monotone contexts. In fact, when

we attach a context of the form $J' \to E \leftarrow J$ to a state $J \to G$, we need the union of J and J' inside E to be exactly all free ports of E. We will call such contexts *net contexts*.



Lemma 2.6.23 (Combinable net context)**.** *Let $J \to G$ be a state which admits a non-silent transition $J \to F \leftarrow J$. Let $J \to E \leftarrow J'$ be a net context. Then $J_\cap = J \cap_E J'$ contains $\bullet$ iff $J \to E \leftarrow J'$ is combinable with $J \to F \leftarrow J$. We call* combinable net context *a net context that contains the free port of the rule addition in its interface.*

Proof. Cospan combinations are defined in Definition 2.3.6. If the cospans are combinable, we know the combination is of the form $J' \to F' \leftarrow J'$, and all four squares are pushouts. We use the narrowing label of $J \to F \leftarrow J$ to get the diagram on the left:



By constructing pullbacks X of $\bullet \to E \leftarrow J'$ and the pullback X' of $F^- \to E' \leftarrow F'$, we get a label $X \to X' \leftarrow X$ which is a narrowing label for $\bullet \to F^- \leftarrow \bullet$, which means it is itself that label. Remember in interaction graphs, that label is minimal with respect to narrowing.

On the other hand, if J' contains $\bullet$, we have the diagram above on the right. By constructing pushouts, one can fill the diagram with the narrowing $\bullet \to F^- \leftarrow \bullet$ to $J \to F \leftarrow J$ that we know to exist, and then by constructing the pushout of $J' \leftarrow \bullet \to F^-$, one gets a graph F' such that $J' \to F' \leftarrow J'$ is the wished cospan combination. □

We can then show, by diagram chasing, that if $J \to E \leftarrow J'$ is combinable with $J \to F \leftarrow J$, then it is possible to build the union $\overline{G}$ of G and E along J and the union $\overline{H}$ of H and E along J in order to have an action from one to the other labeled $C\langle J \to F \leftarrow J\rangle = J' \to F' \leftarrow J'$, where F' is the union of J' and F along $J_\cap$.

We can now build a Structural Operational Semantics for such interaction net systems.

Theorem 2.6.24 (SOS for simply wired interaction graphs)**.** *Let $\mathcal{S} = (\Lambda, \mathcal{R})$ be a simply wired interaction graph system (or with unique partners). Lets consider $\mathcal{T} = (\Lambda, \mathcal{R} \cup \mathcal{R}')$ where $\mathcal{R}'$ are the half-rules described in Proposition 2.6.22.*

We consider acceptable labeled transitions in $\mathcal{T}$ to be either silent transitions yielded by a rule in $\mathcal{R}$ or "usual" transitions yielded from a rule in $\mathcal{R}'$.

Then the Structural Operational Semantics of $\mathcal{T}$ is given in Table 2.3.

(We remind that $A \cup_X B$ is a notation for the pushout of $A \leftarrow X \rightarrow B$, which corresponds here to the union where both images of X are identified).

- **Atomic Actions**

$$\frac{}{(I \to \alpha) \xrightarrow{I \to F \leftarrow I} (I \to R)} \quad \text{where} \quad \begin{array}{l}(L \leftarrow I \to R) \in \mathcal{R}' \\ L = \alpha_i \bowtie \beta_j \\ F = I \cup_{\{\bullet\}} \beta \quad \textit{i.e.} \quad \beta_j = \bullet\end{array}$$

- **Combinable Net Contextualization**

$$\frac{(J \to G) \xrightarrow{J \to F \leftarrow J} (J \to H)}{\overline{J} \to G \cup_J E \xrightarrow{C\langle J \to F \leftarrow J \rangle} \overline{J} \to H \cup_J E} \quad \text{where} \quad \begin{array}{l}J = \mathrm{fp}(G) \\ C = J \to E \leftarrow \overline{J} \text{ is a combinable} \\ \quad \text{net context for } J \to F \leftarrow J\end{array}$$

- **Composition**

$$\frac{t = (J \to G) \xrightarrow{J \to F \leftarrow J} (J \to H) \qquad t' = (J' \to G') \xrightarrow{J' \to F' \leftarrow J'} (J' \to H') \qquad J \leftarrow X \to J'}{\bar{t} = (\overline{J} \to G \cup_X G') \xrightarrow{\overline{J} \to \overline{J} \leftarrow \overline{J}} (\overline{J} \to H \cup_X H')}$$

$$\text{where} \quad \begin{array}{l}t \text{ and } t' \text{are } \tau\text{-compatible} \\ X \text{ contains } \bullet \\ \overline{J} = (J \leftarrow X \to J') \setminus X\end{array}$$

Table 2.3: Axioms and rules of the SOSBC-system for simply wired interaction graphs.

2.7 Conclusion

We haven't proved Theorem 2.6.24. The two differences with the SOS of Table 2.2 is the special nature of atomic actions (the considered interfaces are now minimal) and the lack of interface narrowing rule.

This formulation is given for intuition more than the result itself, as it is not very useful in practice. The labels are complicated, but most of all, it requires the δ and γ cells: it relies on the fact that a rule can be split in two in such a way that the connection between the two halves is made exactly in one node. It is true, as Lafont showed, that it is sufficient to have or to add the δ and γ cells and the two reflexive rules without changing the expressivity of the system to obtain such a system. Almost.

Lafont's splitting property does not for instance consider the case when the two halves are disconnected. It is still possible to add a dummy connecting port that would remain floating around. One should just be careful.

For our purpose, we see a much bigger problem. We are looking to compare systems and there expressivity power, also systems not having δ and γ-cells. We cannot just add them and pretend we have accomplished the comparison. Moreover, in Chapter 4, we prove a universality result: there is a system $\mathcal{U}$ in which any interaction net system can be encoded. Even ones not having these cells already. It is therefore not possible to use this operational semantics to compare a system with its encoding in the universal system. This is why we don't insist on the proofs of this section.

We have made a first step towards a proper SOS semantics for graph transformation systems, by introducing SOS rules not only for a specific graph transformation system, but by presenting a method for synthesizing such rules for arbitrary systems. This will hopefully lead to a better understanding of the nature of SOS semantics in general.

While earlier work on automatic derivation of labeled transition systems, pioneered by [36], focused on the derivation of labeled transitions such that the resulting bisimilarity is a congruence, we here concentrated on synthesizing SOS rules, obtaining a compositional operational semantics.

Composition rules for process calculi, such as CCS or the π-calculus, where at most two processes interact in a well-defined manner, can usually be stated quite concisely, whereas our composition rule is surprisingly complex. In future work we want to investigate under which conditions the label of the composed step can be determined from the labels of the interacting graphs by a simpler procedure. We regard this work as a first step towards a SOS semantics for graph transformation, however in order to obtain a simpler and more straightforward presentation it might be necessary to impose certain constraints on the rules, such as restrictions on the left-hand or right-hand side.

A natural question to ask is how the SOS semantics for a specific calculus or graph transformation system would look like, for instance for the encoding of CSS into graph transformation studied in [6]. One of the synthesized rules would certainly correspond to the CCS communication rule, which states that $P \xrightarrow{a} P'$ and $Q \xrightarrow{\overline{a}} Q'$ imply $P \mid Q \xrightarrow{\tau} P' \mid Q'$. On the other hand we would also generate many other rules, for instance a rule where P in addition borrows the output prefix of Q and Q borrows the input prefix of P. Our SOS rules allow the composition of such labeled transitions, since we accept arbitrary overlaps of the left-hand side and the two graphs to be composed. Hence the full SOS semantics would be somewhat large and unwieldy and difficult to represent. Pruning the synthesized rules to a minimal set is a direction of future work.

Our focus is quite different than that of work on process calculi defining syntactic conditions on the rules in such a way that the resulting bisimilarity is a congruence (compare with the de Simone format [14] and the `tyft/tyxt`-format [27]). Instead,

bisimilarity on labeled transition systems derived via the Borrowed Context technique is automatically a congruence [19], and here we synthesize the SOS rules that are already given in the work on rule formats.

Earlier work has established graph transformation as a formalism into which many process calculi can be encoded or "compiled". This becomes apparent in Milner's work on bigraphs [42, 43] and there are several papers giving a graphical syntax for processes, such as [8, 22] or the well-known interaction nets [32]. Even closer to our work are papers that deal with SOS semantics for graphs such as [10, 28]. In both papers however SOS rules are given a priori, similar to the work on rule formats cited above, while here they are synthesized. A predecessor paper of the current work is [2] which deals with the composition and decomposition of derivations without explicitly focusing on SOS rules, but by combining entire Borrowed Context diagrams. In [52] compositionality for graph transformation is obtained by decomposing rules into subrules, a view that is somehow dual to ours: we do not decompose rules, but the graph to be rewritten. Finally [9] shows how label derivation can be addressed in double categories and explores connections to the tile model [23]. There are similarities to our approach, however [9] works in a different categorical setting and does not investigate composition of LTS steps.

This diversity of work on compositionality in semantics of interactive systems reflects the richness of the subject. Attracted by the simplicity of structural operational semantics, we have developed basic results for a simplified account of the Borrowed Context technique. Even though we have missed the mark of a proper SOS format for interaction nets in general – since composition of labels depends on the involved states – we believe that our results are interesting in themselves and for a better understanding of operational semantics for graph transformation.

Chapter 3

Concurrent Interaction Nets

In this chapter, we study the expressivity of the concurrent extensions of interaction nets. A general definition of a labeled transition having partly failed, we consider here the commonly used approach of *encodability*. What it means exactly is described in Section 3.1.

We first deal with the general expressivity of interaction nets by giving a INS that encodes the π-calculus (Sec. 3.2). Finally, we proceed with the core of the chapter: comparing different concurrent extensions among themselves. Section 3.3 summarizes the encodability and separability results, that are then given in detail: 3.4 show how to encode any interaction net system in a multiport one; 3.5 shows how to express rule ambiguity with multiwires; in 3.6, we show that multirules alone are not enough to replace multiwires; finally, a weak separation result of multiport from multiwires is given in 3.7.

The concepts for this chapter are introduced in the paragraphs about textual notation of interaction nets (p. 7), observational equivalence (p. 11) and expressivity (p. 12).

3.1 (Textual) interaction nets

In order to make this chapter self-contained, we redefine everything about concurrent extensions of interaction nets that is needed to follow the results, but in the framework of the textual representation. From now on, we will say *interaction nets* for their textual version as well as their graphical version, and will use the two epithets only when it will be necessary to stress them. A reader who feels confident with the way textual and graphical representations are similar can skip this section all together.

Nets. In the following, we fix a enumerable infinite set of *ports*, ranged over by lowercase Latin letters. We write $\tilde{x}$ to denote a finite sequence of ports $x_1, \ldots, x_n$ such that every port appears at most twice in the sequence; n is said to be the *length* of $\tilde{x}$. If ports appear at most once, we say that $\tilde{x}$ is *repetition-free*.

Definition 3.1.1 (Net)**.** An *alphabet* is a pair $\Sigma = (|\Sigma|, \deg)$, where $|\Sigma|$ is a set and $\deg : |\Sigma| \to \mathbb{N}$ is the *degree function*.

A *cell*, or *agent*, on the alphabet Σ is an expression of the form $\alpha(\tilde{x})$, where $\alpha \in |\Sigma|$ and $\tilde{x}$ is of length $\deg(\alpha)$.

A *k-connector*, or *multiwire*, is a multiset of cardinality k of ports, containing at most two occurrences of every port, denoted by $[\tilde{x}]$. A 2-connector is called a *simple wire*.

A *net* on an alphabet Σ is a finite multiset of connectors and agents on Σ in which every port appears at most twice. A net is *simply-wired* if it contains no multiwires, but only simple ones.

The set of *free ports* of a net μ, denoted by $\mathrm{fp}(\mu)$, is the set of ports appearing exactly once in μ. The ports appearing twice in a net are called *bound*. We identify any two nets which may be obtained one from the other by an injective renaming of their bound ports (this is *α-equivalence*).

We denote by $\mu\{y/x\}$ the net μ in which the only free occurrence of x is replaced by y. The notation is extended to sequences (*i.e.*, $\mu\{\tilde{y}/\tilde{x}\}$) with the obvious meaning.

Definition 3.1.2 (Juxtaposition)**.** Given two nets μ, ν, we denote by $\mu \mid \nu$ the net obtained by renaming (using α-equivalence) the bound ports of μ and ν so that the two nets have no bound name in common, and by taking then the standard multiset union.

Note that unlike usual process calculi, the symbol $\mid$ is not part of the language but rather an operation defined on nets. It is obviously commutative and has the empty net, denoted by 0, as neutral element. It is not associative in general; however, for $\mu \mid (\nu \mid \rho)$ and $(\mu \mid \nu) \mid \rho$ to be equal, it is enough that $\mathrm{fp}(\mu) \cap \mathrm{fp}(\nu) \cap \mathrm{fp}(\rho) = \emptyset$. More in general, if $\mu_1, \ldots, \mu_n$ are such that, for all pairwise distinct i, j, k, $\mathrm{fp}(\mu_i) \cap \mathrm{fp}(\mu_j) \cap \mathrm{fp}(\mu_k) = \emptyset$, then the expression $\mu_1 \mid \cdots \mid \mu_n$, also denoted by $\prod_{k=1}^{n} \mu_k$, is not ambiguous. Such a notation will always be used under this assumption in the sequel.

In the sequel, by *congruence* on nets we mean an equivalence relation $\sim$ such that $\mu \sim \nu$ implies that for every ρ, $\rho \mid \mu \sim \rho \mid \nu$.

Definition 3.1.3 (Structural congruence)**.** *Structural congruence* $\equiv$ is the smallest congruence on nets satisfying the following:

0-*connector:*	$\mu \mid []$	$\equiv$	μ	
Fusion:	$[\tilde{x}, a] \mid [a, \tilde{y}]$	$\equiv$	$[\tilde{x}, \tilde{y}]$	
Wire:	$\mu \mid [a, x]$	$\equiv$	$\mu\{x/a\}$	if $a \in \mathrm{fp}(\mu)$

A net is *totally disconnected* if it contains no connector and no bound port. Conversely, a *wiring* is a net consisting solely of connectors. We deem *maximal* a connector which shares no name with any other connector. The following is easy to check:

Lemma 3.1.4 (Expanded form)**.** *For every net μ, there exist unique μ_0 and ω such that $\mu \equiv \mu_0 \mid \omega$, where μ_0 is totally disconnected and ω is a wiring of maximal connectors containing all the free ports of μ.*

In fact, it corresponds exactly to the description of a graphical net into cells and wires. We will also need a special property on nets:

Definition 3.1.5 (Symmetric net)**.** A net μ is *symmetric* if $\mu \equiv \mu\{\tilde{x}'/\tilde{x}, \tilde{x}/\tilde{x}'\}$ where $\tilde{x}, \tilde{x}'$ are the free ports of μ. x_i and x'_i are said to be *exchanged by the symmetry.*

Interaction. We fix two infinite sequences of ports $(p_i)_{i\in\mathbb{N}}$ and $(q_i)_{i\in\mathbb{N}}$, which we suppose to never occur in nets.

Definition 3.1.6 (Interaction scheme)**.** An *interaction scheme* on an alphabet Σ is a partial function $\bowtie$ from $|\Sigma| \times \mathbb{N} \times |\Sigma| \times \mathbb{N} \times \mathbb{N}^*$ to nets on Σ such that, where all free ports are reserved:

- if (α, i, β, j, k) is in the domain of $\bowtie$, then $1 \leq i \leq m = \deg \alpha$ and $1 \leq j \leq n = \deg \beta$ and it is denoted $\alpha_i \lhd k \rhd \beta_j$. If $\alpha = \beta$ and $i = j$ the scheme is called a *self-rule*, or *reflexive rule*;
- if defined, the image of (α, i, β, j, k) is denoted by $\alpha_i \lhd k \rhd \beta_j$ and its free ports are exactly $p_1, \ldots, p_{i-1}, p_{i+1}, \ldots, p_m, q_1, \ldots, q_{j-1}, q_{j+1}, \ldots, q_n$;
- furthermore, we require, for any $\alpha \neq \beta$ or $i \neq j$, that $\beta_j \lhd k \rhd \alpha_i$ be also defined and equal to the net obtained from $\alpha_i \lhd k \rhd \beta_j$ by exchanging the ports p_1 and q_1, p_2 and q_2, and so on.
- if $\alpha_i \lhd k \rhd \alpha_i \equiv \alpha_i \lhd k \rhd \alpha_i\{\tilde{q}/\tilde{p}, \tilde{p}/\tilde{q}\}$, the self-rule is called *symmetric.* Otherwise it is *asymmetric.*

A projection (α, i, β, j) of $\bowtie$ is said to be *simple* if for all $k, k' \in \mathbb{N}^*$, $(\alpha, i, \beta, j, k) \in \mathtt{Dom}(\bowtie)$ and $(\alpha, i, \beta, j, k') \in \mathtt{Dom}(\bowtie)$ implies $k = k'$ and if $\alpha = \beta$ and $i = j$ the (unique) self-rule is symmetric. We can, without loss of generality, consider that $\alpha_i \lhd k \rhd \beta_j$ is defined only if $\forall k' \leq k$, $\alpha_i \lhd k' \rhd \beta_j$ is. So a scheme is simple if for all (α, i, β, j, k) in the domain of $\bowtie$, $k = 1$, and $\bowtie (\alpha, i, \alpha, i, 1)$, if defined, is symmetric.

Definition 3.1.7. An *interaction net system* (INS) is a triple $\mathcal{S} = (\Sigma_\mathcal{S}, \bowtie_\mathcal{S}, \mathcal{O}_\mathcal{S})$ where $\Sigma_\mathcal{S}$ is an alphabet, $\bowtie_\mathcal{S}$ is an interaction scheme on $\Sigma_\mathcal{S}$ and $\mathcal{O}_\mathcal{S} \subseteq |\Sigma_\mathcal{S}| \times \mathbb{N} \times |\Sigma_\mathcal{S}| \times \mathbb{N} \times \mathbb{N}$ is non-empty and such that $(\alpha, i, \beta, j, k) \in \mathcal{O}_\mathcal{S}$ implies that $(\beta, j, \alpha, i, k) \in \mathcal{O}_\mathcal{S}$ and that $\mathtt{max}\{n \mid \bowtie (\alpha, i, \beta, j, n) \text{ is defined}\} = l > 0$ and $1 \leq k \leq l$. Subscripts are omitted when clear from the context.

Definition 3.1.8 (Reduction)**.** The *reduction relation* $\to_\mathcal{S}$ of an INS $\mathcal{S}$ is defined as follows:

$$\frac{\alpha_i \lhd k \rhd \beta_j \text{ defined}}{\alpha(\tilde{x}) \mid \beta(\tilde{y}) \mid [x_i, y_j, z] \;\to_\mathcal{S}\; \alpha_i \lhd k \rhd \beta_j\{\tilde{x}/\tilde{p}, \tilde{y}/\tilde{q}\} \mid [z]}\ \text{INTERACTION}$$

$$\frac{\mu \to_\mathcal{S} \mu'}{\mu \mid \nu \to_\mathcal{S} \mu' \mid \nu}\ \text{CONTEXT} \qquad \frac{\mu \equiv \mu' \quad \mu' \to_\mathcal{S} \nu' \quad \nu' \equiv \nu}{\mu \to_\mathcal{S} \nu}\ \text{STRUCT}$$

We denote by $\to_{\mathcal{S}}^*$ the reflexive-transitive closure of $\to_{\mathcal{S}}$. A net structurally congruent to the net on the left side of the INTERACTION rule is called an (α_i, β_j)-*active pair*, denoted $\alpha_i \overset{k}{\bowtie} \beta_j$. Clearly, $\mu \to_{\mathcal{S}} \nu$ only if some (α_i, β_j)-active pair is reduced using the k-th rule for (α, i, β, j). When we need to specify it, we write $\mu \xrightarrow{\alpha_i\beta_j}_k \nu$.

In a INS $\mathcal{S}$, given $\alpha \in |\Sigma_{\mathcal{S}}|$, we say that the i-th port of α is *principal* if $\alpha_i \lhd k \rhd \beta_j$ is defined for some $\beta \in |\Sigma|$ and $k \in \mathbb{N}^*$. Otherwise, it is called *auxiliary*.

A connector containing two occurrences of the same port is said to be *vicious*. We say that a net μ *diverges* if there is an infinite reduction path starting from it, or if it reduces to a net containing a vicious connector.

To improve readability, it is convenient to assume principal ports to be always the "leftmost" in the list of ports of a cell, and to use the notation $\alpha(x_1, \ldots, x_m; y_1, \ldots, y_n)$ for a cell whose symbol α is of degree $m + n$ and has m principal ports. If all ports are principal, the semicolon is omitted.

In practice, when defining an interaction net system, it is convenient to specify the interaction scheme directly by giving rewriting rules of the form

$$\alpha(\tilde{x}) \mid \beta(\tilde{y}) \to_k \nu$$

where $\tilde{x}, \tilde{y}$ are repetition-free, $x_i = y_j = z$ for some i, j, and $\mathrm{fp}(\nu) = \{\tilde{x}, \tilde{y}\} \setminus \{z\}$. It is then intended that $\alpha_i \lhd k \rhd \beta_j$ is defined and equal to $\nu\{\tilde{p}/\tilde{x}, \tilde{q}/\tilde{y}\}$. It also automatically defines $\beta_j \lhd k \rhd \alpha_i$.

An INS can have one or few of the following properties. It can be:

- *uniport* if all its symbols have exactly one principal port; otherwise, it is called *multiport* (MP).
- *simply-wired* (SW), if all reduction rules introduce simply-wired nets (in that case, one restricts to simply wired nets); otherwise, it is called *multiwired* (MW).
- *simple* (SR), if all of its projections are simple; otherwise, it is said *multirule* (MR).

Lafont interaction nets are uniport, simply-wired, simple interaction nets. Any other combination of these adjectives denote a kind of non-deterministic interaction net system. The meaning and use of the set $\mathcal{O}_{\mathcal{S}}$ (the *observable rules*) will be made clear later.

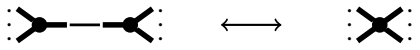
Graphical representations. We wish, for the sake of exhaustivity, to make precise the relation between textual and graphical representations. Following the convention that all principal ports are listed first in the cell, a cell

$$\alpha(a_1, \ldots, a_n; a_{n+1}, \ldots, a_m)$$

(a) A cell α (b) Uniport cell (c) A one-cell (d) A 5-connector

Figure 3.1: Cells and connectors

of degree m with n principal ports is pictured as in Fig. 3.1(a). Cells with only one principal port, such as $\beta(a; a_1, \ldots, a_n)$, or cells of degree 1 such as $\gamma(a)$ are represented more conveniently as in Fig. 3.1(b) and Fig. 3.1(c), respectively. Connectors are represented as Fig. 3.1(d). We add a simple arc between any two occurrences of a port. To represent graphically structural congruence, we use the graphical equality as a rewriting rule in both ways:



Behavioral equivalence. In the following, we fix an arbitrary INS.

Definition 3.1.9 (Residue, interreduction)**.** Given an active pair ϕ of a net μ and a reduction $\mu \to \mu'$ reducing an active pair ψ, we have two possibilities: either ϕ and ψ share a cell (the extreme case being $\phi = \psi$), or they are disjoint. In the first case, ϕ has no residue in μ'; in the second case, the cells of ϕ are left untouched by the reduction, and μ' contains an active pair ϕ' which is "the same" as ϕ. This is its *residue* in μ'. The notion of residue is extended to reductions of arbitrary length in the obvious way.

Let μ be a net, and let F be a set of active pairs of μ. We say that a reduction $\mu \to^* \mu'$ is *F-legal* if it reduces no active pair of F nor any of their residues.

Let μ, ν be two nets, and let F, G be the set of *all* of their respective active pairs. An *interreduction* of $\mu \mid \nu$ is a reduction which is $F \cup G$-legal (juxtaposition may create active pairs not in $F \cup G$; this is why the definition is sensible.).

Definition 3.1.10 (Barbed bisimilarity)**.** Let $\mathcal{S}$ be an INS. We say that a reduction step $\mu \xrightarrow{\alpha_i \beta_j}_k \nu$ is *observable* if $(\alpha, i, \beta, j, k) \in \mathcal{O}_{\mathcal{S}}$.

We write $\mu{\downarrow_x}$ if there exists a net o such that $\mathrm{fp}(\mu) \cap \mathrm{fp}(o) = \{x\}$ and an interreduction of $\mu \mid o$ containing an observable step. We write $\mu{\Downarrow_x}$ if $\mu \to^* \mu'{\downarrow_x}$ and we say that o is an *observer* of x in μ.

Let $\mathcal{S}, \mathcal{T}$ be two INSs. A *barbed $(\mathcal{S}, \mathcal{T})$-bisimulation* is a binary relation $\mathcal{B} \subseteq \mathcal{S} \times \mathcal{T}$ on nets s.t. $\mathcal{B}(\mu, \nu)$ implies

- for every port x, $\mu{\downarrow_x}$ implies $\nu{\Downarrow_x}$ and $\nu{\downarrow_x}$ implies $\mu{\Downarrow_x}$;
- $\mu \to_{\mathcal{S}} \mu'$ implies that there exists ν' s.t. $\nu \to^*_{\mathcal{T}} \nu'$ and $\mathcal{B}(\mu', \nu')$;
- $\nu \to_{\mathcal{T}} \nu'$ implies that there exists μ' s.t. $\mu \to^*_{\mathcal{S}} \mu'$ and $\mathcal{B}(\mu', \nu')$.

If there exists a barbed $(\mathcal{S},\mathcal{T})$-bisimulation $\mathcal{B}$ such that $\mathcal{B}(\mu,\nu)$, we say that μ and ν are barbed bisimilar and write $\mu\ {}_{\mathcal{S}}\dot{\approx}_{\mathcal{T}}\ \nu$ (subscripts are dropped in a clear context).

Barbed congruence for $\mathcal{S}$, denoted by $\simeq^{\mathrm{c}}_{\mathcal{S}}$, is the greatest congruence contained in ${}_{\mathcal{S}}\dot{\approx}_{\mathcal{S}}$.

The above definition of barb may be applied to standard name-passing calculi: if the only reduction rule (i/o synchronization) is considered observable, we obtain the usual barbs. The concept of interreduction is necessary to guarantee that the observable step does not come from active pairs already present in μ or, worse, in the observer o.

Usually, barbed congruence is defined to be the greatest congruence included in barbed bisimilarity. The two definitions are known to coincide in the π-calculus. We ignore whether this is the case for net systems. In this work we chose this definition because it allows the use of coinductive arguments, which will be exploited for proving the validity of our encodings (3.2.5 and 3.4.14).

Definition 3.1.11 (Translation)**.** Let $\mathcal{S},\mathcal{T}$ be INSs. A *translation* from $\mathcal{S}$ to $\mathcal{T}$ is a map $[\![\cdot]\!]$ from nets of $\mathcal{S}$ to nets of $\mathcal{T}$ s.t., for all nets μ,μ' of $\mathcal{S}$:

Homomorphism: $[\![0]\!]=0$ and $[\![\mu\mid\mu']\!]=[\![\mu]\!]\mid[\![\mu']\!]$;

Port invariance: for every net μ of $\mathcal{S}$, $\mathrm{fp}([\![\mu]\!])=\mathrm{fp}(\mu)$, and if $\mu=\nu\{\tilde{x}/\tilde{p}\}$, we have $[\![\mu]\!]=[\![\nu]\!]\{\tilde{x}/\tilde{p}\}$;

Operational correspondence: we have the following two properties:

- $\mu\to_{\mathcal{S}}\mu'$ implies $[\![\mu]\!]\to^*_{\mathcal{T}}\simeq^{\mathrm{c}}_{\mathcal{T}}[\![\mu']\!]$;
- $[\![\mu]\!]\to^*_{\mathcal{T}}\nu$ implies $\exists$ a net μ' of $\mathcal{S}$ s.t. $\mu\to^*_{\mathcal{S}}\mu'$ and $\nu\to^*_{\mathcal{T}}\simeq^{\mathrm{c}}_{\mathcal{T}}[\![\mu']\!]$;

Bisimulation: $\mu\ {}_{\mathcal{S}}\dot{\approx}_{\mathcal{T}}\ [\![\mu]\!]$.

A translation *does not introduce divergence* if, whenever $[\![\mu]\!]$ diverges, μ diverges.

The bisimulation condition corresponds to what Gorla [26] calls "success sensitiveness", in that it excludes trivial translations which would otherwise be validated by the other three conditions (such as an encoding mapping every net with free ports $x_1,\ldots,x_n$ to the net $[x_1],\ldots,[x_n]$). Furthermore, bisimulation implies the adequacy and relative completeness of translations with respect to barbed congruence (of the respective systems):

adequacy: $[\![\mu]\!]\simeq^{\mathrm{c}}_{\mathcal{T}}[\![\mu']\!]$ implies $\mu\simeq^{\mathrm{c}}_{\mathcal{S}}\mu'$;

relative completeness: $\mu\simeq^{\mathrm{c}}_{\mathcal{S}}\mu'$ implies $\forall$ net ρ of $\mathcal{S}$, $[\![\rho]\!]\mid[\![\mu]\!]\ {}_{\mathcal{T}}\dot{\approx}_{\mathcal{T}}\ [\![\rho]\!]\mid[\![\mu']\!]$.

This is a consequence of the (easy to verify) fact that, for any three INSs $\mathcal{S},\mathcal{T},\mathcal{U}$, $\mu\ {}_{\mathcal{S}}\dot{\approx}_{\mathcal{T}}\ \nu$ and $\nu\ {}_{\mathcal{T}}\dot{\approx}_{\mathcal{U}}\ \rho$ implies $\mu\ {}_{\mathcal{S}}\dot{\approx}_{\mathcal{T}}\ \rho$.

We will see in Section 3.4.2 that problems occur when we wish to translate connectors. In such cases, we will need to loosen up the definition of translation.

The reader might be disturbed by the parametricity of observability. Two answers can be given to that. First: the more general the definition, the more general the result. This is the scientists approach. More practically, it is always possible to consider a default definition of barbs, that is, that all interactions are observable. Such a choice makes every free principal port a barb. This is not completely senseless: if we regard the solos calculus [35] as a textual interaction net system, and we deem its only interaction rule observable, we obtain the usual barbs. It is only when considering encodings that smaller sets of observable rules need to be considered. A target system of a translation will most probably have more rules that the source language, and it is natural to consider some of those to be *administrative*: they serve an organizing purpose, more that a computing one – reorganizing connections, duplicating parts of code, garbage collection... These interactions should most certainly not be considered observable.

3.2 Encoding the pi-calculus

We consider the synchronous, monadic π-calculus, as defined for instance in [54]. For convenience, we use the same letters to range over π-calculus names and ports. Processes are defined by

$$\begin{array}{rcl@{\qquad\qquad}rcl} \kappa & ::= & \overline{x}(y) \;\big|\; \pi & \pi & ::= & \overline{x}y \;\big|\; x(y) \\ P, Q & ::= & M \;\big|\; P \mid Q \;\big|\; \nu(z)P \;\big|\; !\kappa.P & M, N & ::= & \mathbf{0} \;\big|\; \pi.P \;\big|\; M + N \end{array}$$

We denote by $\mathtt{fn}P$ the set of free names of the process P.

The restriction on the form of replicated processes is known not to diminish the expressive power of the calculus (see, for instance, Exercise 2.2.30 of [54]) and allows us to discard the equation $!P \equiv P \mid !P$ in favor of reduction axioms such as $x(z).P + M \mid !\overline{x}(z).Q \rightarrow \nu(z)(P \mid Q) \mid !\overline{x}(z).Q$, determining how replicated prefixes interact with sums and other replicated prefixes, in the obvious way. This will greatly simplify our encoding.

We do not consider match (or mismatch) prefixes. These may be encoded at the cost of further technical complications which we believe are unnecessary to show the expressiveness of interaction net systems.

Some notation. We will introduce the alphabet of the interaction net system encoding the π-calculus gradually, together with the encoding itself, which we denote by $(\!|\cdot|\!)$. The encoding will have the property that $\mathrm{fp}((\!|P|\!)) = \mathtt{fn}P$. To help us maintain this equality, we will need the notation

$$(x)\mu \;=\; \begin{cases} \mu & \text{if } x \in \mathrm{fp}(\mu), \\ \mu \mid [x] & \text{if } x \notin \mathrm{fp}(\mu), \end{cases}$$

where μ is an arbitrary net. Another convenient notation for wirings is the following:

$$[\![\tilde{x}^1, \ldots, \tilde{x}^m]\!] \;=\; [x_1^1, \ldots, x_1^m] \mid \cdots \mid [x_n^1, \ldots, x_n^m],$$

where we implicitly assume that $\tilde{x}^1, \ldots, \tilde{x}^m$ have all length n. Furthermore, if μ, ν are arbitrary nets and if $\tilde{x}$ spans $\mathrm{fp}(\mu) \cap \mathrm{fp}(\nu)$, we define

$$\mu \parallel \nu \;=\; \mu\{\tilde{a}/\tilde{x}\} \mid \nu\{\tilde{b}/\tilde{x}\} \mid [\![\tilde{x}, \tilde{a}, \tilde{b}]\!].$$

Unlike juxtaposition, this parallel composition is strictly associative (and is still commutative and has neutral element 0).

Null process, parallel composition and name restriction. We start with the agent-free part of the encoding:

$$\begin{aligned} (\!|\mathbf{0}|\!) &= 0, \\ (\!|P \mid Q|\!) &= (\!|P|\!) \parallel (\!|Q|\!), \\ (\!|\nu(x)P|\!) &= [x] \mid (x)(\!|P|\!). \end{aligned}$$

Guarded sums. We introduce a family of symbols $(\sigma_n^m)_{m,n\in\mathbb{N}}$, of degree $3(m+n)$ and with $m+n$ principal ports. A guarded sum containing m input prefixes with subjects $\tilde{w}$ and bound names $\tilde{z}$, together with n output prefixes with subjects $\tilde{x}$ and sending names $\tilde{y}$, will be encoded by a net of the form $\mu = \sigma_n^m(\tilde{w}, \tilde{x}; \tilde{z}, \tilde{y}, \tilde{c}, \tilde{d}) \mid \nu$, with ν containing the encodings of the continuations, located at $\tilde{c}$ for the inputs and $\tilde{d}$ for the outputs. Of course there may be other σ_k^h cells in ν; the one shown corresponds to the top-level prefixes. If $\mu' = \sigma_{n'}^{m'}(\tilde{w}', \tilde{x}'; \tilde{z}', \tilde{y}', \tilde{c}', \tilde{d}') \mid \nu'$ is another net of this form, we define

$$\mu \oplus \nu \;=\; \sigma_{n+n'}^{m+m'}(\tilde{w}, \tilde{w}', \tilde{x}, \tilde{x}'; \tilde{z}, \tilde{z}', \tilde{y}, \tilde{y}', \tilde{c}, \tilde{c}', \tilde{d}, \tilde{d}') \mid (\nu \parallel \nu'),$$

which is still of the same form. Then, we define

$$(\!|M + N|\!) = (\!|M|\!) \oplus (\!|N|\!).$$

Prefixing. To encode continuations, we need to introduce two families of symbols $(\gamma_n)_{n\in\mathbb{N}}$ (multiplexors) and $(\beta_n)_{n\in\mathbb{N}}$ (blockers), all having one principal port and such that $\deg(\gamma_n) = \deg(\beta_n) = n+1$. We will also need two symbols φ (unblocker) and ε (eraser), both of degree 1. All these symbols are deemed *administrative*. Their

interaction rules, also called administrative, are the following:

$$\begin{aligned}
\gamma_n(x;\tilde{y}) \mid \gamma_n(x;\tilde{z}) &\to [\![\tilde{y},\tilde{z}]\!] \\
\gamma_n(x;y_1,\ldots,y_n) \mid \varphi(x) &\to \varphi(y_1) \mid \cdots \mid \varphi(y_n) \\
\gamma_n(x;y_1,\ldots,y_n) \mid \varepsilon(x) &\to \varepsilon(y_1) \mid \cdots \mid \varepsilon(y_n) \\
\beta_n(x;y_1,\ldots,y_n) \mid \varphi(x) &\to [y_1,\ldots,y_n] \\
\beta_n(x;y_1,\ldots,y_n) \mid \varepsilon(x) &\to [y_1] \mid \cdots \mid [y_n]
\end{aligned}$$

Let now μ be a net. By 3.1.4, we have $\mu \equiv \mu_0 \mid [\tilde{x}_1] \mid \cdots \mid [\tilde{x}_n]$, where μ_0 is a totally disconnected net and $\tilde{x}_1,\ldots,\tilde{x}_n$ contain all the free ports of μ (and therefore all the ports appearing in μ_0). Let $k_1,\ldots,k_n$ be the lengths of $\tilde{x}_1,\ldots,\tilde{x}_n$, respectively, and let c be a port not appearing free in μ. We define the net

$$c.\mu \;=\; \gamma_n(c;\tilde{d}) \mid \beta_{k_1}(d_1;\tilde{x}_1) \mid \cdots \mid \beta_{k_n}(d_n;\tilde{x}_n) \mid \mu_0.$$

Obviously, $\mathrm{fp}(c.\mu) = \mathrm{fp}(\mu) \cup \{c\}$, and it is easy to check the following:

Lemma 3.2.1. *The net $c.\mu$ is normal and has the following properties:*

1. $c.\mu \mid \varphi(c) \;\to^*\; \mu$;
2. $c.\mu \mid \varepsilon(c) \;\to^*\; [y_1] \mid \cdots \mid [y_m] \mid \zeta$, *where $y_1,\ldots,y_m$ are the free ports of μ and ζ is a normal net with no free ports.*

Moreover, both of the above reductions consist entirely of administrative steps.

We may now give the encoding of single prefixes:

$$\begin{aligned}
(\!|\overline{x}y.P|\!) &= \sigma_1^0(a;b,c) \mid c.(\!|P|\!)\{a'/x,b'/y\} \mid [x,a,a'] \mid [y,b,b'], \\
(\!|x(y).P|\!) &= \sigma_0^1(a;y,c) \mid (y)(c.(\!|P|\!))\{a'/x\} \mid [x,a,a'].
\end{aligned}$$

The interaction rule of σ_n^m cells is the following:

$$\begin{aligned}
\sigma_n^{m+1}(\tilde{w},\tilde{x};\tilde{z},\tilde{y},\tilde{c},\tilde{d}) \mid \sigma_{n'+1}^{m'}(\tilde{w}',\tilde{x}';\tilde{z}',\tilde{y}',\tilde{c}',\tilde{d}') \;\to\; & [z_i,y_j'] \mid \varphi(c_i) \mid \varphi(d_j') \\
& \mid \varepsilon(\tilde{c}_-) \mid \varepsilon(\tilde{d}) \mid \varepsilon(\tilde{c}') \mid \varepsilon(\tilde{d}'_-) \\
& \mid K(\tilde{w}_-,\tilde{x},\tilde{z}_-,\tilde{y},\tilde{w}',\tilde{x}'_-,\tilde{z}',\tilde{y}'_-),
\end{aligned}$$

where $w_i = x_j'$ and we used the notations $K(\tilde{a}) = [a_1] \mid \cdots \mid [a_k]$ and $\varepsilon(\tilde{a}) = \varepsilon(a_1) \mid \cdots \mid \varepsilon(a_k)$, and $\tilde{w}_-, \tilde{z}_-, \tilde{c}_-$ (resp. $\tilde{x}'_-, \tilde{y}'_-, \tilde{d}'_-$) stand for $\tilde{w},\tilde{z},\tilde{c}$ without x_i, z_i, c_i (resp. $\tilde{x}',\tilde{y}',\tilde{d}'$ without x_j', y_j', d_j'). The intuition is that the name sent (y_j') and the binder (z_i) are connected, while two φ cells are dispatched to "unblock" the selected continuations. The other continuations are erased, and all the other ports are "discarded" (this is the function of the 1-connectors in the right hand side).

Replication. We introduce a further administrative cell, the duplicator δ, of degree 3 and with 1 principal port, which interacts with itself and with any cell $\alpha \neq \delta$ of

degree n as follows (these reductions too are considered administrative):

$$\begin{aligned}\delta(x;y,z) \mid \delta(x;u,v) &\to [y,u] \mid [z,v]\\ \delta(x_i;y,z) \mid \alpha(\tilde{x}) &\to \alpha(\tilde{a}) \mid \alpha(\tilde{b}) \mid \delta(x_1;a_1,b_1) \mid \cdots \mid \delta(x_n;a_n,b_n),\end{aligned}$$

where, in the second rule, $a_i = y$ and $b_i = z$ and there is no cell $\delta(x_i;a_i,b_i)$.

Given a net μ and a port $c \notin \mathrm{fp}(\mu)$, if $\tilde{x}$ is a sequence of length n spanning $\mathrm{fp}(\mu)$, we define

$$c!_{\tilde{x}}\mu \;=\; \gamma_{n+1}(c;a,\tilde{x}) \mid a.\mu,$$

which has only one free port c. This is a sort "duplicable package" containing μ. To "extract" μ from $c!_{\tilde{x}}\mu$, we use the net $D_n(c;\tilde{z}) = \gamma_{n+1}(c;b,\tilde{z}) \mid \varphi(b)$.

Lemma 3.2.2. *The net $c!_{\tilde{x}}\mu$ is normal and satisfies the following:*

1. $c!_{\tilde{x}}\mu \mid \delta(c;d,e) \;\to^*\; d!_{\tilde{x}}\mu \mid e!_{\tilde{x}}\mu$;
2. $c!_{\tilde{x}}\mu \mid D_n(c;\tilde{z}) \;\to^*\; \mu\{\tilde{z}/\tilde{x}\}$.

Moreover, both of the above reductions consist entirely of administrative steps.

We introduce three families of symbols $(!\iota_n)_{n\in\mathbb{N}}$, $(!o_n)_{n\in\mathbb{N}}$ and $(!o^b_n)_{n\in\mathbb{N}}$, representing replicated input, replicated output and replicated bound output, respectively, of degree $n+3$, $n+4$ and $n+3$, respectively, all with 1 principal port. The encoding of replicated prefixes is as follows:

$$\begin{aligned}(\!|!\overline{x}y.P|\!) &= !o_k(a;a',y,\tilde{u},c) \mid c!_{x,y,\tilde{u}}(x)(y)\mu \mid [x,a,a']\\ (\!|!x(y).P|\!) &= !\iota_k(a;a',\tilde{u},c) \mid c!_{x,y,\tilde{u}}(x)(y)\mu \mid [x,a,a']\\ (\!|!\overline{x}(y).P|\!) &= !o^b_k(a;a',\tilde{u},c) \mid c!_{x,y,\tilde{u}}(x)(y)\mu \mid [x,a,a']\end{aligned}$$

In all three cases, $\tilde{u}$ spans $\mathtt{fn}P \setminus \{x,y\}$ and is supposed to be of length k. The extra port a' is a copy of the subject name needed to spawn a new copy of the prefix. The interaction rules with summation cells are as follows:

$$\begin{aligned}\sigma^{m+1}_n(\tilde{w},\tilde{x};\tilde{z},\tilde{y},\tilde{c},\tilde{d}) \mid !o_k(w_i;w',y',\tilde{u},e) \to\; & [z_i,y',b,b'] \mid \varphi(c_i) \mid \varepsilon(\tilde{c}_-) \mid \varepsilon(\tilde{d}) \mid \delta(e;f,g)\\ & \mid D_{k+2}(g;a',b',\tilde{u}') \mid !o_k(a;a'',b,\tilde{u}'',f)\\ & \mid [w',a,a',a''] \mid [\![\tilde{u},\tilde{u}',\tilde{u}'']\!]\\ \sigma^{m+1}_n(\tilde{w},\tilde{x};\tilde{z},\tilde{y},\tilde{c},\tilde{d}) \mid !o^b_k(w_i;w',\tilde{u},e) \to\; & [z_i,b'] \mid \varphi(c_i) \mid \varepsilon(\tilde{c}_-) \mid \varepsilon(\tilde{d}) \mid \delta(e;f,g)\\ & \mid D_{k+2}(g;a',b',\tilde{u}') \mid !o^b_k(a;a'',\tilde{u}'',f)\\ & \mid [w',a,a',a''] \mid [\![\tilde{u},\tilde{u}',\tilde{u}'']\!]\\ \sigma^{m}_{n+1}(\tilde{w},\tilde{x};\tilde{z},\tilde{y},\tilde{c},\tilde{d}) \mid !\iota_k(x_i;x',\tilde{u},e) \to\; & [y_i,b'] \mid \varphi(d_i) \mid \varepsilon(\tilde{c}_-) \mid \varepsilon(\tilde{d}) \mid \delta(e;f,g)\\ & \mid D_{k+2}(g;a',b',\tilde{u}') \mid !\iota_k(a;a'',\tilde{u}'',f)\\ & \mid [x',a,a',a''] \mid [\![\tilde{u},\tilde{u}',\tilde{u}'']\!]\end{aligned}$$

where we used the same notational convention as above for the "missing" ports on the right hand side. Basically, together with the usual behavior of summations, a replicated prefix agent also creates a copy of itself, and dispatches a duplicator to make two copies of the continuation; one of these copies is kept as it is, the other is "extracted" by means of the net D_{k+2}.

A similar, but expectedly twice as complicated behavior is obtained when these agents interact with themselves:

$$\begin{aligned}
!\iota_h(x;w,\tilde{u},c) \mid !o_k(x;z,y,\tilde{v},d) \to\ & [y,b,b',b''] \mid \delta(c;f,g) \mid \delta(d;i,j) \mid D_{h+2}(g;a',b',\tilde{u}') \\
& \mid \iota_h(a;a'',\tilde{u}'',f) \mid D_{k+2}(j;e',b'',\tilde{v}') \mid !o_k(e;e'',b,\tilde{v}'',i) \\
& \mid [w,a,a',a''] \mid [z,e,e',e''] \mid [\![\tilde{u},\tilde{u}',\tilde{u}'']\!] \mid [\![\tilde{v},\tilde{v}',\tilde{v}'']\!] \\
!\iota_h(x;w,\tilde{u},c) \mid !o^b_k(x;z,\tilde{v},d) \to\ & [b,b'] \mid \delta(c;f,g) \mid \delta(d;i,j) \mid D_{h+2}(g;a',b,\tilde{u}') \\
& \mid \iota_h(a;a'',\tilde{u}'',f) \mid D_{k+2}(j;e',b',\tilde{v}') \mid !o^b_k(e;e'',\tilde{v}'',i) \\
& \mid [w,a,a',a''] \mid [z,e,e',e''] \mid [\![\tilde{u},\tilde{u}',\tilde{u}'']\!] \mid [\![\tilde{v},\tilde{v}',\tilde{v}'']\!]
\end{aligned}$$

Validity of the encoding. The INS encoding the π-calculus, which we call Π, is obtained by taking the system introduced in the previous section and considering as observable all non-administrative reductions.

Below, we write $\mu \to_{\mathrm{a}}^* \mu'$ if the reduction is purely administrative. The following lemma states that administrative reductions are deterministic.

Lemma 3.2.3. *Let $\mu \to_{\mathrm{a}}^* \mu''$, and let $\mu \to^* \mu'$ be another, arbitrary reduction, of length n. Then, there exists ν such that $\mu'' \to^* \nu$ and $\mu' \to_{\mathrm{a}}^* \nu$. Moreover, the length of the reduction from μ'' to ν is at most n.*

Proof. Consider $\mu \to_{\mathrm{a}} \mu_2$ and $\mu \to \mu_1$. Since administrative active pairs do not overlap with other active pairs, we immediately have a net ν_1 such that $\mu_2 \to \nu_1$ and $\mu_1 \to_{\mathrm{a}} \nu_1$, unless $\mu_1 = \mu_2$, in which case confluence is trivial (this is why we do not get that the length of $\mu'' \to^* \nu$ is *exactly* n, but may be shorter). The result then follows by a standard diagram chasing argument, using an induction on n. □

Lemma 3.2.4. *1. For every π-calculus process P, the active pairs of $(\!|P|\!)$ are in bijection with the one-step reductions of P.*

2. Let $P \to P'$. Then, $(\!|P|\!) \to \mu \to_{\mathrm{a}}^* (\!|P'|\!) \mid \zeta$, where the first reduction step is not administrative, and ζ is a normal net with no free ports.

Proof. Point 1 is proved by induction on P. Point 2 is a tedious but straightforward verification, using Lemmas 3.2.1 and 3.2.2. The fact that we discarded the rule $!P \equiv P \mid !P$ is essential; the lemma would be false otherwise. □

Theorem 3.2.5. *For every π-calculus process P, we have:*

completeness: *$P \to^* P'$ implies $(\!|P|\!) \to^* \simeq^{\mathrm{c}}_{\Pi} (\!|P'|\!)$;*

soundness: $(\!|P|\!) \to^* \mu'$ *implies that there exists* P' *s.t.* $P \to^* P'$ *and* $\mu' \to^* \simeq^{\mathrm{c}}_{\Pi} (\!|P'|\!)$.

Proof. Completeness is a straightforward consequence of 3.2.4 and the fact, which is not hard to prove, that $P \equiv Q$ implies $(\!|P|\!) \simeq^{\mathrm{c}}_{\Pi} (\!|Q|\!)$. The fact that $\simeq^{\mathrm{c}}_{\Pi}$ is a barbed bisimulation is used here. Soundness is proved by induction on the length n of the reduction $(\!|P|\!) \to^* \mu'$. If $n = 0$, the result is trivial. Otherwise, we have $(\!|P|\!) \to \mu_1 \to^* \mu'$, and the length of the reduction from μ_1 to μ' is strictly smaller than n. By point 1 of 3.2.4, there exists P_1 s.t. $P \to P_1$. By point 2, the step $(\!|P|\!) \to \mu_1$ is not administrative, and we have $\mu_1 \to^*_{\mathrm{a}} (\!|P_1|\!) \mid \zeta = \mu''$. We are in position to apply 3.2.3, which gives us a net ν such that $\mu' \to^*_{\mathrm{a}} \nu$ and $\mu'' \to^* \nu$, and the latter reduction is still of length strictly smaller than n. But ζ is normal, so we actually have $\nu = \nu_0 \mid \zeta$ and $(\!|P_1|\!) \to^* \nu_0$, and we may apply the induction hypothesis, which gives us P' such that $P \to P_1 \to^* P'$ and $\nu_0 \to^* \simeq^{\mathrm{c}}_{\Pi} (\!|P'|\!)$. So $\mu' \to^* \simeq^{\mathrm{c}}_{\Pi} (\!|P'|\!) \mid \zeta$, and we may conclude because $\zeta \simeq^{\mathrm{c}}_{\Pi} 0$, a fact that is immediate to show for any net with no free ports. □

3.3 Comparing interaction net extensions

In this section, we study in detail the relative expressivity of different concurrent extensions of interaction nets, namely interaction nets with *multiple rules*, *multiwires*, *multiport cells* and all there combinations. We remind the abreviations:

MR Nets with multiple rules, a.k.a. *multirule systems*.

MW Rules introduce multiwires for any k, , a.k.a. *multiwired systems*.

MP Nets with multiport cells, a.k.a. *multiport systems*.

In this section, unless clearly specified, the kind of system is strict: MR means the system has multirules but simple wires and simple cells only; MW allows connectors of any size but simple rules and simple cells; MP has multiport cells but simple rules and simple wires. We express combinations explicitly:

MWR Multiple rules of which some introduce multiwires.

MPR Nets with multiport cells and multiple rules.

MPW Nets with multiport cells and rules introducing multiwires.

MPWR Nets with multiport cells, multiwires and multiple rules, alias *general nets*.

In the following, we will usually denote the source interaction net system by $\mathcal{S}$ and the target system by $\mathcal{T}$ or $\mathcal{S}^*$, unless specified otherwise. We will denote the encoding of a net μ by $[\![\mu]\!]$, since which encoding is used will be clear from the context. In graphical representations, interaction is written with an arrow $\to$, reduction by a starred arrow $\to^*$ and the encoding by a squigarrow $\mu \rightsquigarrow \nu$ means the same as $[\![\mu]\!] = \nu$. In diagrams, arrows between interaction net frameworks means relative expressiveness: for $\mathcal{S}, \mathcal{S}'$

among MR,MW,MP,MWR,MPR,MPW,MPWR, $\mathcal{S} \to \mathcal{S}'$ means there is an encoding from $\mathcal{S}$ to $\mathcal{S}'$ or that $\mathcal{S}$ is a subsystem of $\mathcal{S}'$ ($\mathcal{S} \subseteq \mathcal{S}'$).

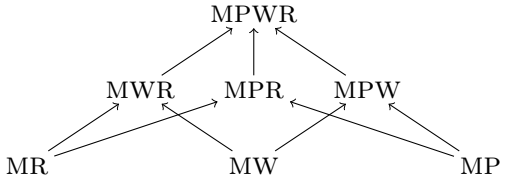
Before we continue, we need to make an important remark. Active pairs composed of two cells of same label are qualified of *reflexive* as are the rules defined on reflexive pairs (see Def. 3.1.6). The place to give to such rules is interesting. They never appear in actual calculi: Lafont forbids them in his first definition of interaction nets [32] before allowing them for universality purposes in [33]; even in a wish to make π-calculus symmetric, Sangiorgi still distinguishes input from output prefixes [53]. It is not clear if allowing them brings any expressivity. Lafont for instance gives an alternative to his combinators which have no reflexive rule. The question of their expressivity cannot be answered strictly in terms of encodings, as a cell that allows a reflexive rule must most probably, for any encoding, need a reflexive rule to trigger the simulation. On the other hand, as we will see, any multirule system can be encoded into a system with a unique (non-symmetric) reflexive rule, so reflexive rules seem to have a great expressivity power.

Non-symmetric reflexive rules are problematic. Such rules are by nature *multirules*, since there are at least two ways of replacing an active pair by the asymmetric right-hand-side. Therefore, one has to be carefull when manipulating reflexive rules, especially when triying to "get rif of multirules". This is why we avoid using them for the first encoding. This is not really a problem, since we later on show how to take care of them. Nevertheless, we have to take them into account in the separation between multiports and multiwires-multirules: the result is only valid in the absence of asymmetric reflexive rules. We still believe that the separation is true in their presence, just are not able to show it yet.

Let us now present the relative expressivity results.

3.3.1 Encodability

We can consider the lattice of interaction net concurrent extensions as shown in the following diagram, where arrows represent inclusion:



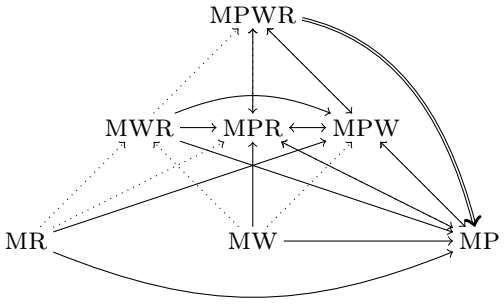
Or if the reader prefers a table:

	MR	MW	MWR	MP	MPR	MPW	MPWR
MR	$=$		$\subseteq$		$\subseteq$		$\subseteq$
MW		$=$	$\subseteq$			$\subseteq$	$\subseteq$
MWR			$=$				$\subseteq$
MP				$=$	$\subseteq$	$\subseteq$	$\subseteq$
MPR					$=$		$\subseteq$
MPW						$=$	$\subseteq$
MPWR							$=$

where inclusion is given from a framework in the title column into systems in the title row. The reader can notice the unnatural position of MWR in the table. It is explained by the first result, given in the following paragraph.

Multiports can alone express rule ambiguity and multiple connections The first encoding result is quite powerful. It is, what one would call a strike.

We give, for any general interaction net system $\mathcal{S}$ a simple, simply wired multiport system $\mathcal{T}$ which is multiport and has the exact same behavior. Therefore, we obtain straight away an equivalence between MP, MPWR, MPW and MPR, since encoding into MP means encoding into any of its extensions, and encoding from MPWR is encoding from any of its subsystems. We also get, as special cases, encodings from MWR, MW and MR into MP.

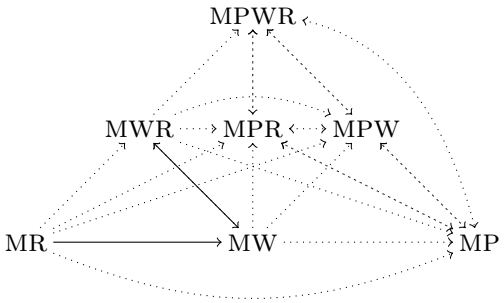


Or in the table version:

	MR	MW	MWR	MP	MPR	MPW	MPWR
MR	$=$		$\subseteq$	✓	$\subseteq$	✓	$\subseteq$
MW		$=$	$\subseteq$	✓	✓	$\subseteq$	$\subseteq$
MWR			$=$	✓	✓	✓	$\subseteq$
MP				$=$	$\subseteq$	$\subseteq$	$\subseteq$
MPR				✓	$=$	✓	$\subseteq$
MPW				✓	✓	$=$	$\subseteq$
MPWR				✓	✓	✓	$=$

The encoding is quite complicated to express but the ideas underlining it are quite simple. We have been enlightened anyhow in Chapter 1 thanks to conflict graphs (see section 1.3.3) about the fact that multiports seem to be a combination of multirules with multiwiring. We are going to make it clearer in Section 3.4, by first giving intuitions on the two encodings separately on systems with simple cells, and then only combine the two adding the multiport dimension.

Multiwires can control rule ambiguity We continue with the study of the relation between multiwires and multiple rules. We give here an encoding of MWR into MW, which bears of course an encoding from MR to MW. Moreover, if the source language has no self-rules, it is possible to define a translation into a MW without self-rules. We will also see how we can use the same idea to go straight from MR to MP.



Or in the table version:

	MR	MW	MWR	MP	MPR	MPW	MPWR
MR	=	✓	⊆	✓	⊆	✓	⊆
MW		=	⊆	✓	✓	⊆	⊆
MWR		✓	=	✓	✓	✓	⊆
MP				=	⊆	⊆	⊆
MPR				✓	=	✓	⊆
MPW				✓	✓	=	⊆
MPWR				✓	✓	✓	=

3.3.2 Separation

Multirules alone do not give concurrency The first separation result is rather strong. It says that it is not possible to use multirules to express multiport cells or rules that use multiwires. It is a question of observability. In plain multirule systems, any non-deterministic behavior is internal: it has no consequence on the interface. If a port is weakly observable, it stays so. Which is not at all the case in systems which allow multiple connections or multiport cells.

	MR	MW	MWR	MP	MPR	MPW	MPWR
MR	$=$	✓	$\subset$	✓	$\subset$	✓	$\subset$
MW	×	$=$	$\subseteq$	✓	✓	$\subseteq$	$\subseteq$
MWR	×	✓	$=$	✓	✓	✓	$\subseteq$
MP	×			$=$	$\subseteq$	$\subseteq$	$\subseteq$
MPR	×			✓	$=$	✓	$\subseteq$
MPW	×			✓	✓	$=$	$\subseteq$
MPWR	×			✓	✓	✓	$=$

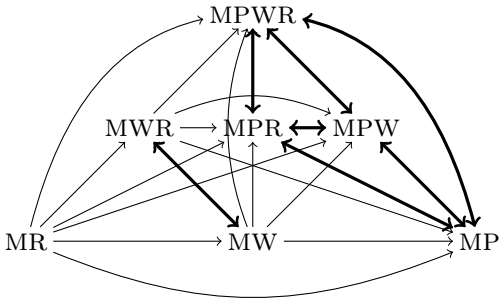
Comparing multiport and multiwire concurrency In this paragraph, we give a weak separation result from MP to MWR, which automatically gives us separations from MP to MW, MP to MR and all the equivalents for MPR, MPW and MPWR. It actually concerns systems without self-rules. There is a series of nets not using self-rules in multiport systems that cannot be translated into MWR with no self-rules without introducing divergence.

It is not convenient to show the absence of arrows in a diagram, so we will just give the table version of what we get as a result of this encoding:

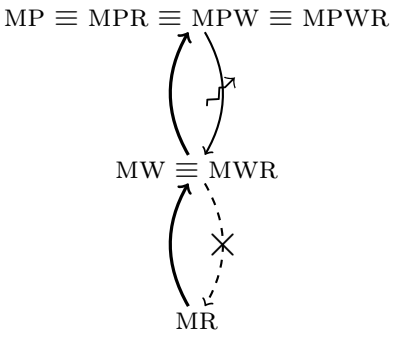
	MR	MW	MWR	MP	MPR	MPW	MPWR
MR	$=$	✓	$\subset$	✓	$\subset$	✓	$\subset$
MW	×	$=$	$\subseteq$	✓	✓	$\subseteq$	$\subseteq$
MWR	×	✓	$=$	✓	✓	✓	$\subseteq$
MP	×	↗	↗	$=$	$\subseteq$	$\subseteq$	$\subseteq$
MPR	×	↗	↗	✓	$=$	✓	$\subseteq$
MPW	×	↗	↗	✓	✓	$=$	$\subseteq$
MPWR	×	↗	↗	✓	✓	✓	$=$

3.3.3 To sum-up

The final diagram for translatability is:



To synthesize, the hierarchy for interaction nets is of the following form:



where $\dashrightarrow\!\!\!\!\!\!\times$ mean no translation exists and $\longrightarrow\!\!\!\!\!\!\nearrow$ that any translation introduces divergence (in the restricted case of self-rule-free systems).

We dedicate now a section to each encoding and separability result discussed above.

3.4 Multiports can alone express rule ambiguity and connectors

The encoding from general nets to MP is technical, so we split it in two[1]. We first deal with systems which have simple cells but multiwires and/or multiple rules. Then, we engage the conversation on what some of the difficulties occur when upgrading to the case of an already multiport source language (*i.e.* a general interaction net system). The last subsection is the technical one, giving and proving the actual encoding.

3.4.1 Uniports, but multwires and/or multiple rules

We first show how to transform multiwires into simply wired nets. The idea is inspired from *communication zones* of Ehrahrd and Laurent's encoding of π-calculus into differential nets [16]. We then show how to get rid of multirules. We then combine the two in a rather straightforward way.

Multiwires as communication zones Let us take a MW system $\mathcal{S}$, thus with simple rules and simple cells. Let α, β, γ be three labels of $\mathcal{S}$ that form a net μ by being all attached by there principal port on a same bound 3-connector w.

For each cell of $\mathcal{S}$, we have a cell $\overline{\alpha}^1$ which represents α, has same arity and coarity. The subscript means it has one copy of α's principal cell. In fact, we have, for all $n \in \mathbb{N}^*$ a cell $\overline{\alpha}^n$ which represents α with n copies of its principal port. Intuitively, a cell of the form $\overline{\alpha}^n(a_1, \ldots, a_n)$ represents the net $\alpha(z; \tilde{x}) \mid [z, a_1, \ldots, a_n]$, *i.e.*, an α-cell whose principal port is attached to a $(n+1)$-connector. In other words, we "embed" the connectors in the cells themselves, modifying their degree if necessary.

[1] We understand we miss the whole point of the unique encoding, but the result still holds. Only the intuitions are "split".

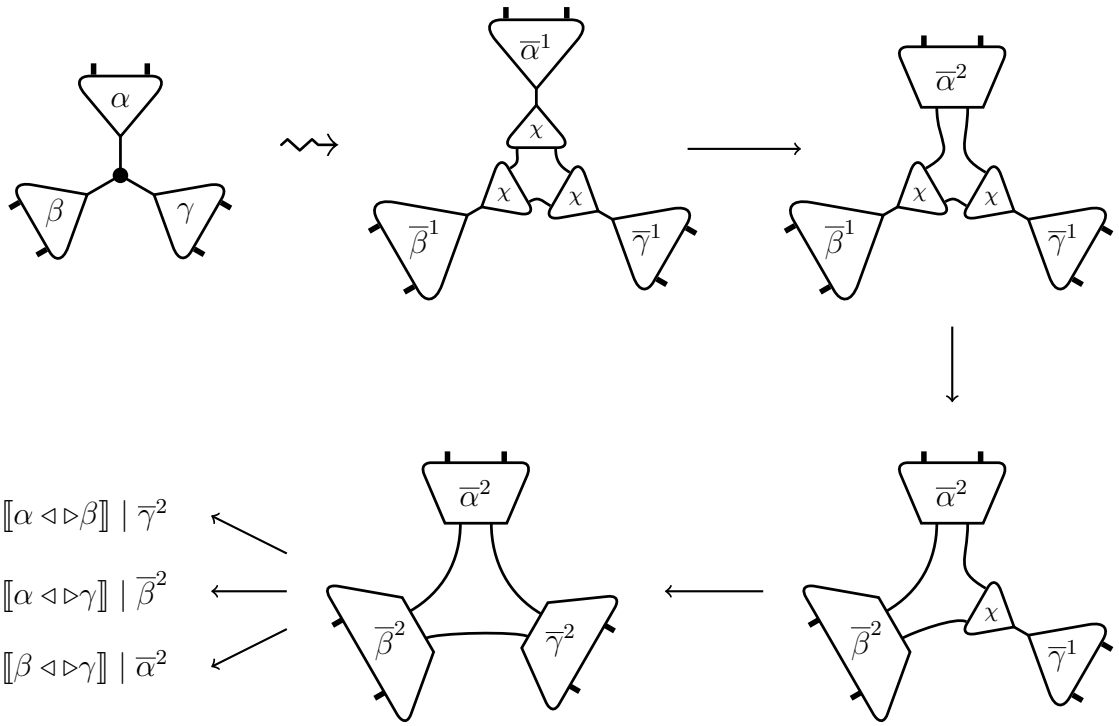


Figure 3.2: The translation and simulation of a net with a 3-connector.

We add to the language a label χ, of degree 3 of which one only port is principal. When a χ meets a "normal" cell c, it duplicates c's principal port and connects the two copies to χ's auxiliary ports. The 3-connector is replaced by three χ-cells, connected 2-by-2 by their auxiliary ports. The translation and the ongoing simulation(s) is shown in Figure 3.2. The final three interactions correspond to the real interactions that are simulated. $[\![\alpha \triangleleft \triangleright \beta]\!] \mid \overline{\gamma}^2$ for instance corresponds to the reduction of $\alpha \bowtie \beta$ in the original net, so γ is left alone. It therefore contains the translation of the RHS of that rule. If it exists. Otherwise, it is not defined and the non-active pair is encoded as a non-active pair.

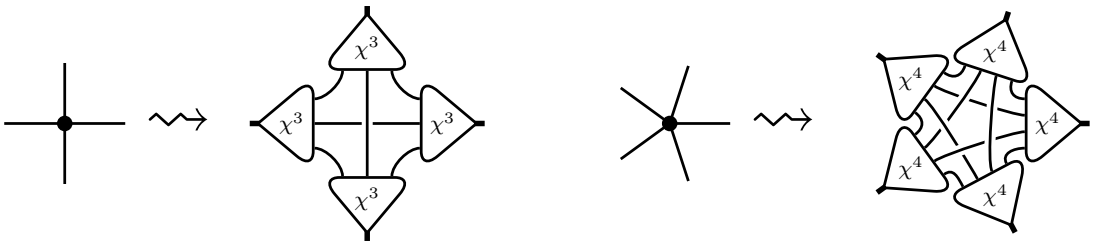
We can see that even if the rule is defined, the simulation of the interaction in the encoding net not necessarily brings to the encoding of the result of the interaction in the source net. In the example of the active pair $\alpha \bowtie \beta$, the cell $\overline{\gamma}^2$ is not exactly the encoding of γ. It is possible, in a simple case like this one, to send a message to $\overline{\gamma}^2$ and "re-transform" it into a $\overline{\gamma}$. It shall be done by sending on all principal ports of $\overline{\alpha}^2$ and $\overline{\beta}^2$ that are not involved in the interaction a zerocell that erases useless principal ports it meets (and erases χ-cells all together on the way). Since we work up-to bisimilarity, we do not need this useless yet aesthetic addition.

We can notice here a nice and important feature of this encoding. Or rather of the χ-cells. Let us call ν_1, ν_2, ν_3 and ν_4 the four first nets of the simulation above (in the target system) and μ_1 the result of reducing the active pair $\overline{\alpha}^2{}_1 \bowtie \overline{\beta}^2{}_2$ from ν_4. As we

can see, ν_3 already contains the active pair $\overline{\alpha}^2{}_1 \bowtie \overline{\beta}^2{}_2$, so it can be reduced to a net we call μ'. Well μ' on its turn can be reduced to μ_1 by reducing the residue of the active pair $\overline{\gamma}^1 \bowtie \chi$. In fact, this is true for any reduction involving a χ-cell: it is *independent* from any other interaction, *almost* in the sense of events in event structures.

If a net ν has two reductions $\nu \to \nu_1$ and $\nu \to \nu_2$ of which one is an interaction involving a χ-cell, then there is a net μ s.t. $\nu_1 \to^* \mu$ and $\nu_2 \to^* \mu$. In some sense, ν cannot introduce non-determinism on its own. It is *deterministic*. This will be true of other rules we will add later for χ and we will use this property a lot to show bisimulation results.

We can generalize χ-cells to any arity in order to simulate larger multiwires:

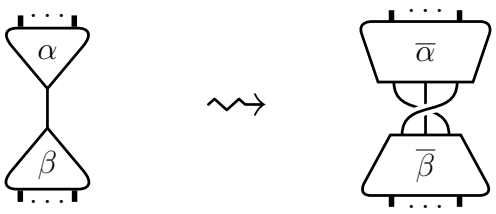


With the general rule that

$$\overline{\alpha}^1(a; b_1, \ldots, b_m) \mid \chi^p(a, x_1, \ldots, x_p) \to \overline{\alpha}^p(x_1, \ldots, x_p; b_1, \ldots, b_m),$$

we can see how the translation works in general. Of course, we are missing some parts of it, but we hope the reader can wait a little for the details.

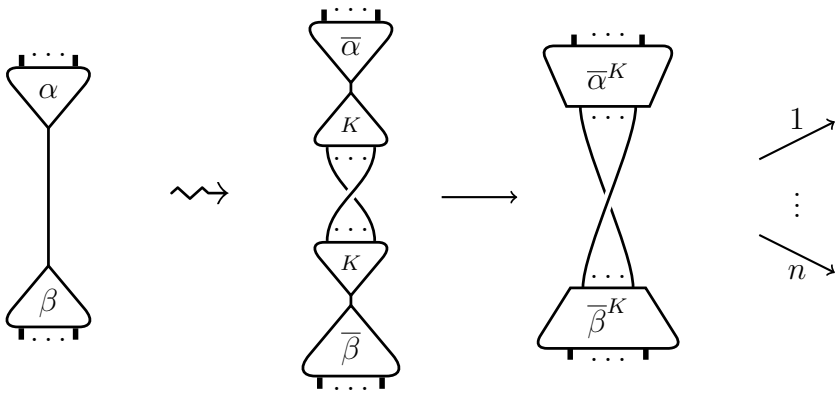
From multiple rules to multiple ports We use of a similar strategy to get rid of multirules. For simplicity, we first consider an interaction system $\mathcal{S}$ with simple cell, simple wires and multiple rules. Moreover none of the self-rules is asymmetric. Imagine two simple cells α, β that have 3 rules for interaction. It suffice to translated each of them into a multiport cell with 3 principal ports connected two by two, where the interaction on the first ports triggers the equivalent of the first rule for $\alpha \bowtie \beta$, the interaction on their second ports triggers the equivalent of the second rule, and so on.



This is clearly not sufficient. First of all, the translation of each cell does not have the same interface as the cell, which is problematic, since if each rule $\alpha \overset{i}{\bowtie} \beta$ is observable, then each of the free ports a, b, c of a net $\overline{\alpha}(a, b, c; \tilde{x})$ will be observable. Also,

the encoding cannot be made *homomorphic*: how to glue two translations together? Second, it could be that α has more rules with another label γ, in which case $\overline{\alpha}$ should have more principal ports. But we cannot know in advance who α is going to be connected to. We solve both problems simultaneously.

Let K be the maximal number of rules for any pair of the source system $\mathcal{S}$ and $n = K(\alpha, \beta)$ the number of rules for $\alpha \bowtie \beta$. We translate each cell by itself (again with an over-lined label) connected by its principal port to the only auxiliary port of a cell k.



A multiwire is decomposed as a *bundle* of K wires each *bearing one rule*. A K-cell splits the principal port of a cell into K ports. A rule is defined for $\overline{\alpha}_i^K \bowtie \overline{\beta}_j^K$ iff $i = j \leq K(\alpha, \beta)$. Then, $\overline{\alpha}_i^K \blacktriangleleft\!\blacktriangleright \overline{\beta}_i^K$ is the net $\alpha \blacktriangleleft\!\blacktriangleright \beta$ in which each cell has been replaced by its encoding.

Now imagine we allow $\mathcal{S}$ to have multiwires. We need to split the multiwire in the same way into K multiwires. This is done by a communication zone in the spirit of the previous section but composed of K-cells. Some examples are given in Figure 3.3.

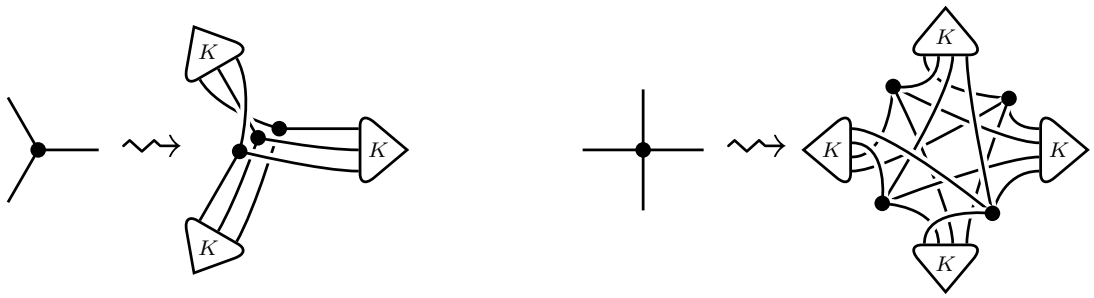


Figure 3.3: Decomposing a multiwire for K rules, $K = 3$ and $K = 4$

Equipped with these K-communication zones, we can describe the translation. Let us take again the simple case of three cells c_1, c_2 and c_3, labeled α, β, γ, connected by a 3-wire. Any cell labeled α is translated by a cell $\overline{\alpha}^1$, and any multiwire $[a_1, \ldots, a_n]$

by a communication zone

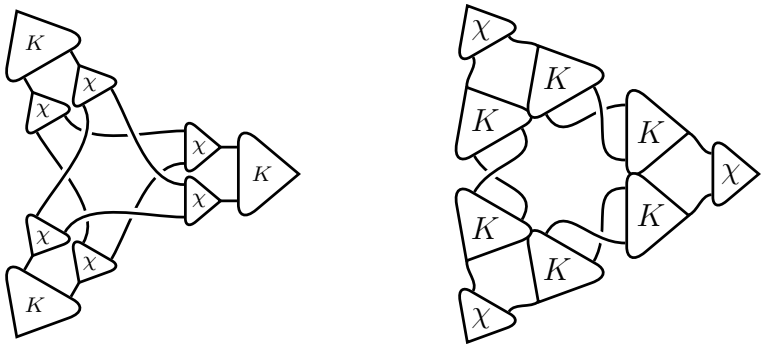
$$K(a_1; u_1^1, \ldots, u_K^1) \mid \ldots \mid K(a_n; u_1^n, \ldots, u_K^n) \mid [u_1^1, \ldots, u_1^n] \mid \ldots \mid [u_K^1, \ldots, u_K^n]$$

The translation of our net is composed of three cells with a K-communication zone. After reducing all (independent) $\alpha \bowtie K$, $\beta \bowtie K$ and $\gamma \bowtie K$, we obtain a net with three cells of coarity K in which all first principal ports of c_1, c_2, c_3 are connected by a 3-wire, all second principal ports also, and so on. The choice of which interaction to trigger corresponds to the simultaneous choice of an active pair of the original net and a number for the rule to use to reduce it.

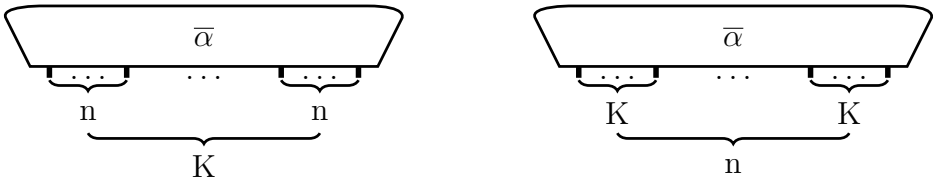
Combining the two From now on, to distinguish the two previous cases of communication zones, we will speak of *χ-zones* and *K-zones*. The combination of the two we give here will be referred to as *general communication zones*, or just *comzones*.

We have seen that the main idea to get rid of multiwires is to encode them into communication zones of χ-cells. We have also seen how to just get rid of multirules even in the presence of multiwires. We can now eliminate both at the same time.

There are two ways of doing so. Transform multiwires into K-zones and replace each multiwire in them by a χ-zone (on the left below), or transform multiwires into χ-zones and replace each wire in them by a K-zone (on the right). A communication zone between 3 ports of a system for which $K = 2$ is one of the following:



A clear difference appears when those communication zones interact with a *normal cell*. K splits principal ports while χ duplicates them. The choice that is made will lead to split ports that are duplicated (as many times as needed from the multiwire) or to duplicated ports that are split:



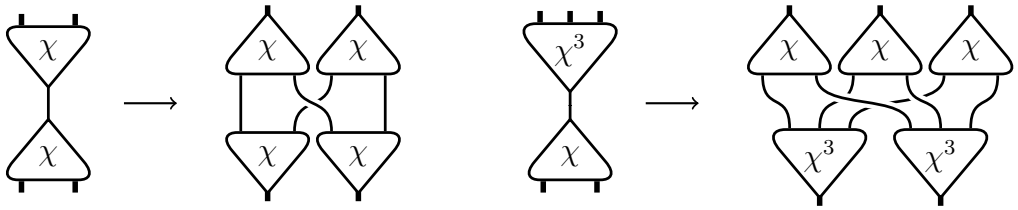
In the first case, the first group of ports represents n copies of the first rule, in the second case, the first group represents the K rules of a single copy. We also have to

keep in mind all the intermediate steps. In the first choice, some rules are copied an other no, while in the second some copies are split and others no.

We can already give a complexity argument in favor of the first choice. A n-connector in a system with K-rules is encoded, in the first version, in a net containing $n(K+1)$ cells, in the second version in a net containing n^2 cells. K being constant for a system while the size of multiwires is unbounded, we are talking liner versus quadratic size. Also, splitting and copying a port of a cell connected to a multiwire of n ports in a system with at most K-rules needs $K+1$ reduction steps in the first choice is made, while in the other, it requires n steps. Here its constant versus linear time. We are not that interested in complexity for this to be a convincing argument. What drives our choice is that the communication zones of the first kind can be combined.

Fusion of wires In the presence of multiwires, it can happen that a port belongs to two of these. In this case, we use the fusion equation of structural congruence (see Definition 3.1.3) that says that two wires in that situation can be considered as a single wire. This situation can appear during reduction. An active pair $\alpha_i \bowtie \beta_j$ has a RHS in which all the ports of the interface belong to multiwires for instance. Imagine a net μ in which all ports of α and β are already connected to multiwires. In its encoding, $\overline{\alpha}$ and $\overline{\beta}$'s ports are all connected to communication zones. Once the reduction for $\alpha_i \bowtie \beta_j$ is simulated, all ports of the interface of that rule belong to two com-zones.

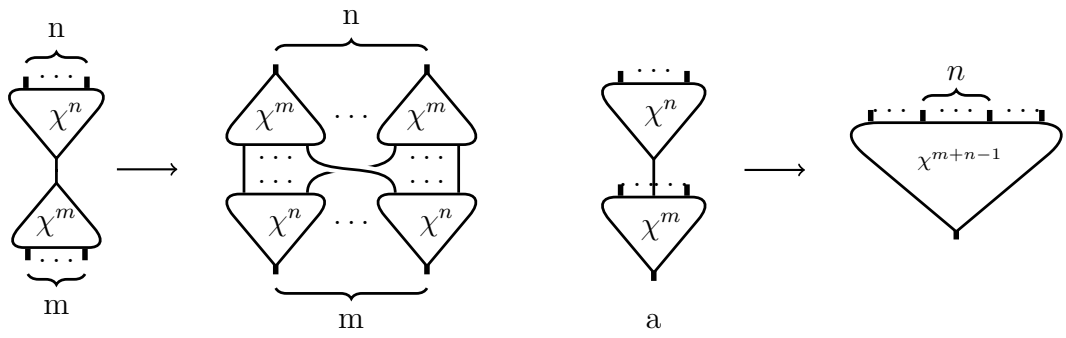
We therefore need to make communication zones have the property of being able to combine. For this, we give rules for active pairs composed of χ-cells. What does a χ-cell represent? It is nothing else than a potential connection between copies of its principal port and all of its auxiliary ports. So what do two χ-cells c_1, c_2 represent? A potential two-by-two connection between all auxiliary port of c_1 and all auxiliary ports of c_2.



Each χ-cell duplicates the other one as many times as its own arity, thus making explicit the two-by-two connections. The general rule is shown in Figure 3.4(a).

The use of the rule above brings to a situation of trees of χ-cells. What do such trees do? They duplicate a port, and then duplicate its copies. It can be achieved in one step, if we add a rule for $\chi_1^m \bowtie \chi_i^n$, shown in Figure 3.4(b), where $i \neq 1$ (making this way all ports of χ-cells principal; for commodity, we draw only the first port of χ-cells as a principal port).

With the two rules, we get what we were looking for, *i.e.* a way to fuse two χ-zones zones together. A simple example of a fusion of χ-zones is shown in Figure 3.5.



(a) Autoduplication (b) Fusion of χ-cells

Figure 3.4: The general rule for χ-cells.

Lemma 3.4.1 (Fusion of χ-zones)**.** *Let $\mathcal{S}$ be a simple system with multiwires ($K = 1$). Let $\mathcal{T}$ be a translation system of $\mathcal{S}$ in which the translation multiwires by χ-zones. For any two wires $w = [a, x_1, \ldots, x_m], w' = a, y_1, \ldots, y_n$ that share a single name, let $v = [x_1, \ldots, x_m, y_1, \ldots, y_n]$ (so $v \equiv w \mid w'$). Then*

$$[\![w \mid w']\!] = [\![w]\!] \mid [\![w']\!] \to^* [\![v]\!]$$

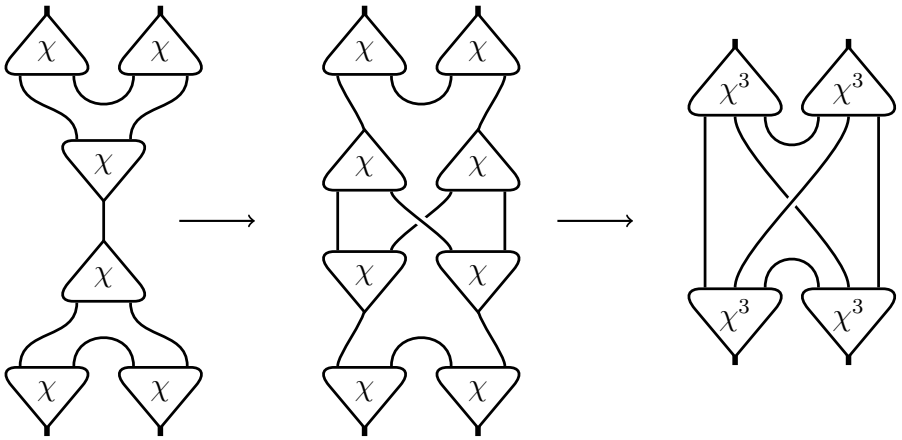
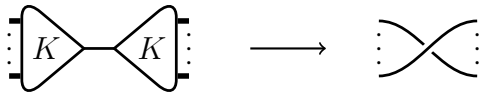


Figure 3.5: Two 3-comzones fuse.

If one considers the more general communication zones (with K-cells), the result stays valid. It is a bit more complicated to write down, but not to visualize. Remember that in such a case, a general comzone is composed of K χ-zones which extremities are shared by K-cells. When two of those K-cells meet on their principal port, the interaction brings together their auxiliary ports two by two.



$$K(a; x_1, \ldots, x_K) \mid K(a; y_1, \ldots, y_K) \quad \to \quad [x_1, y_1] \mid \ldots \mid [x_K, y_K].$$

When two communication zones meet, the K-cells creating the link connect their auxiliary ports yielding K fusion situations between χ-zones, discussed right above. Therefore, the following lemma holds:

Lemma 3.4.2 (Fusion of comzones)**.** *Let $\mathcal{S}$ be a multiport system with multiwires, with multirules of degree at most K. Let $\mathcal{T}$ be a translation system of $\mathcal{S}$ in which the translation multiwires by comzones. For any two wires $w = [a, x_1, \ldots, x_m], w' = a, y_1, \ldots, y_n$ that share a single name, let $v = [x_1, \ldots, x_m, y_1, \ldots, y_n]$ (so $v \equiv w \mid w'$). Then*

$$[\![w \mid w']\!] = [\![w]\!] \mid [\![w']\!] \to^* [\![v]\!]$$

This is the real argument for this choice of communication zones over the other (see p. 111).

What simulation does to the cells Translating the multiwires with external K cells leads to cells $\overline{\alpha}$ in which a partition into K classes $\tilde{p}_1, \ldots, \tilde{p}_K$ of its principal ports is made, such that each port in $\tilde{p}_i$ represents in some sense a possible use of the i-th rule of an active-pair.

In our simple example with three cells c_1, c_2, c_3 (labeled resp. α, β, γ) in a system with at most 2-rules, it means the following. When reducing completely the communication zone in-between the three cells, we obtain cells $c'_1, c'_2 c'_3$, each with 4 principal ports, the first two of which represent the first rule and the second two the second possible rule. So we have a certain number of copies of ports representing a certain *rule number*. The only difficulty is to keep track of which port is a copy of which rule number.

To do so, we will annotate the cell label by a semi-colon separated string of integers that keeps track of all operations made on the principal ports of the cell. $\overline{\alpha}^{1:2:2:11:1}$ means that the cell is the encoding of a α cell from a system for which $K = 5$, in which the principal port is split and the first and last rules have one copy, the second rule and third rule have two copies, the third rule has 11 copies. If the port is not split, we use the notation I as the entire string. The short for $1 : \ldots : 1$ will be K and for $n : \ldots : n$ will be $n \cdot K$.

Definition 3.4.3 (ID of a cell, number of a port)**.** We call *identification* of a cell c, or $\text{ID}(c)$, the string annotating the label of c. Given an ID $a_1 : \ldots : a_K$, also called *K-type*, and an integer $i \leq \Sigma_{j=1}^{K} a_j$, we call *number of i in c*, denoted $\sharp_{\text{ID}(c)}(i)$ or directly $\sharp_c(i)$ the minimal p for which $\Sigma_{j=0}^{p} a_j \geq i$.

In other words, the number of a port is the rule it represents, that we can determine from the cells ID.

We now need to determine rules for active pairs between cells with IDs. Let c, c' two cell labeled respectively $\overline{\alpha}^{ID}$ and $\overline{\beta}^{ID'}$. Then, there is a rule for $\overline{\alpha}_i^{ID} \bowtie \overline{\beta}_j^{ID'}$ iff $\sharp_{ID}(i) = \sharp_{ID'}(j) = n$ and there is a rule number n for the active pair $\alpha \bowtie \beta$ in the original system. The RHS of the rule is *almost* the encoding of the RHS of the original corresponding rule. We just need to bound all the images of the principal ports, and for an even better result, send a message on them that these ports are useless and the connections can be erased.

3.4.2 Some issues with communication zones

We address some precise questions in this paragraph about communication zones. Are they a translation of multiwires? Do we need χ^n-cells for unbounded n? We also discuss the possibility for communication zones to introduce divergence.

Are communication zones translations of wires? Wires have a special nature in nets. In fact, they are the real purpose of the existence of a structural congruence. We have seen already how the fusion equation commands the rules of communication zones. In fact, the situation is even worse. Because ports of cells are numbered, there is no possible way a net containing cells but only simple wires can have the same symmetries as a multiwire, which has all possible permutations on its free ports.

Let us take the case of a simply wired net μ with 3 free ports x, y, z. To be the translation of a 3-connector, it would need to be such that $\mu \equiv \mu\{y/z, z/y\}$. This means that there is a path from x to y iff there is a path from x to z. Since x and y are not connected by a wire, otherwise x and z should too, x is connected to a cell c. The path from x to y passes necessarily through this cell c. So their is a path from another port x' of c to y. But then, there must be a path from x' to z. This process can go on along the entire path going from x to y. When it reaches the cell y is connected to, the only way this last port is also connected to z is by a multiwire, of size at least 3.

The induction step is possible because ports are numbered. A simple illustration of that is that, for any label α of power 3,

$$\alpha(a, x, y) \neq \alpha(a, y, x).$$

A slightly more complex illustration is given in Figure 3.6. There are several ways of encoding a wire $[a, b, c]$ as χ-zones, depending on the original choice of the representant of the multiset. The χ-zones for $[a, b, c]$ is not isomorph to the communication zone representing $[a, c, b]$. This is because of the commutativity of multisets that nets do not have.

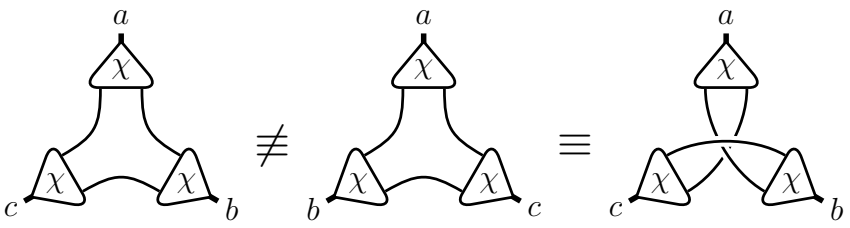


Figure 3.6: Encodings of a same multiwire are not necessarily structurally congruent.

Our encoding is therefore not a translation, since we do not map a multiwire to a net, but rather to a collection of nets. The choice of which net is used for the actual encoding of a given wire is unimportant, as all nets representing a communication zone of a given size n are barbed congruent. Despite this rather important fact, the condition of Definition 3.1.11 still hold. If we consider the encoding of a net to be the set of all the nets that only differ by their encoding of a communication zone, we have

- homomorphism, in which $[\![\mu]\!] \mid [\![\nu]\!]$ means "pick one in each encodings";
- port invariance as equality of sets, where the substitution is made point-wise on every net of the set;
- operational correspondence and bisimulation work for every element of the set of encodings of μ.

We can therefore consider we have an translation *up-to-barbed congruence*.

We prefer to state the result in the following way. For any interaction net system $\mathcal{S}$ (for now, only uniport), there is a simply wired multiport system with simple rules $\mathcal{T}$ such that: for any net μ in $\mathcal{S}$, there is a net ν in $\mathcal{T}$ with the same degree of distribution, which does not depend on the free ports of μ and which has the exact same behavior. Therefore, multiwires and multirules do not extend the operational expressivity of multiport interaction net systems. In other words, in presence of multiport cells, there is no need for multirules and multiwires.

This is not it. Structural congruence on wires brings other problems.

Simple wires It is a little problem that our encoding does not encode simple wires as simple wires. In fact, this means that we cannot encode directly any net but only an *extended version* where ports do not belong to two cells at once. The encoding is therefore *up-to structural congruence*, which is not to bad.

It is possible anyhow to remedy this little problem by adding a simple cell c of arity and coarity 1 that *checks* if there is a wire or not. A cell α is encoded as a cell $\alpha^{I,\dots,I}(z_1,\dots,z_n) \mid c(a_1,z_1) \mid \dots \mid c(a_n,z_n)$. If c meets a K cell, then it just disappears. If not, then it must meet another c-cell. In which case, there interaction provides exactly the encoding of the wire, meaning back-to-back K-cells with their wiring. In order not to complicate the encoding, we do not use this strategy and just use the encoding up-to-structural congruence making all simple wires explicit, using the *wire*

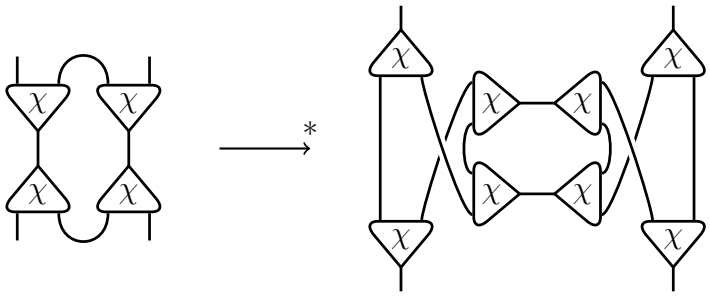


Figure 3.7: A simple divergent net.

equation of structural congruence (Def. 3.1.3) as a right-to-left rewriting rule. In the absence of multirules, no change has to be made to the encoding, as in such a case simple wires are encoded as simple wires.

Do we need χ^n for any n? From the results of Mazza [40] related in Chapter 1, we can guess that in order to encode multiwire systems into simply wired multiport, we will most certainly need a infinite language of cells. It might be possible to find an encoding bypassing that problem, since Mazza's result is only valid for encoding that yield bisimilar embeddings. Our encoding does yield them. It does not mean we cannot try to restrict the "amount of infinite cells".

Generalized χ-cells can be avoided. In fact, they are nothing else than trees of simple χ-cells.

Definition 3.4.4 (χ-tree)**.** A *χ-tree of root x* is a net defined inductively as follows: $[x, z]$ is a χ-tree of root x; if X, X' are χ-trees of root a, a', $\chi(x; a, a') \mid X \mid X'$ is a χ-tree of root x. The free ports of a χ-tree other than its root are called *leaves*; their number is the *arity* of the tree.

We let the reader check that χ-trees behave like generalized χ-cells for both rules above (the second being trivial since connecting the root of a tree to the leave of another is a tree by definition). Later, when we also introduce κ-cells, we will authorize some other structures to represent χ-trees. These cells act like plugs, so a tree that has some leaves plugged by κ-cells still acts as a tree, just of smaller arity.

Divergence of fusion The reader might have notices that we stressed, during all the discussion about fusion, that the wires should share at most one name. It is compatible with the fusion equation of structural congruence (see Def. 3.1.3), which is only valid in that case. Why is that?

Consider the reduction represented in Figure 3.7. It is a (growing in size) infinite reduction. The net originating it appears when two comzones share two or more extremities. Notice that this does not occur for K-zones, but it a property of χ-zones, and thus of general communication zones.

Two comzones share two or more extremities when they are encodings of two connectors that share two or more names. Lafont already considered them as *vicious* (in the case of simple wires), as they are the simplest deadlock in Lafont nets. It is interesting that this encoding treats (at least one kind of) deadlock as livelock.

Since the encoding of multiwires brings such problems, we could have worked up-to structural congruence all the way. We could consider, for any net μ its expanded form (see Definition 3.1.4). The translation of a net would be the translation of its expanded form. This avoids divergence because of cyclic wires, avoids missing some hidden simple wires, but would still not be a map from nets to nets, as the symmetries of multiwires are still a problem. We would also loose homomorphism, even if we could still argue that the parallel distribution is maintained.

3.4.3 Upgrade to multiport source language

If the source system is already multiport, we need to keep track of the splittings and duplications of every principal port in cells. For this, we redefine the ID to be a string of IDs, of length corresponding to the coarity of the cell.

Let $\mathcal{S}$ be a source language which is MPWR with K-rules at most and $\mathcal{T}$ the target language which is MP.

Definition 3.4.5 (ID of a cell, number of a port)**.** Let $K \in \mathbb{N}$. We call *identification of K-types* (ID) a string of the form $W = Z^1, \ldots, Z^n$ where for each $i \leq n$, Z^i is either a string of (strictly positive) natural numbers $a^i_1 : \ldots : a^i_K$, or I. In the first case, $|Z^i| = \Sigma^K_{j=1} a_j$, in the second case $|I| = 1$. Each Z_i is called a *type* of the ID.

Let $N = \Sigma^n_{i=1}|Z^i|$ (the sum of all integers in all substrings of the ID). N is called *coarity* of W.

For any $p \leq N$, the *type* of p in W, denoted $\mathrm{T}_W(p)$, is the least $t \leq n$ s.t. $p \leq \Sigma^t_{j=1}|Z^j|$. The *number of i in W*, denoted $\sharp_W(i)$ is defined as follows:

- if $Z^{\mathrm{T}_c(i)} = I$, then $\sharp_c(i) = I$;
- if $Z^{\mathrm{T}_c(i)} = a_1 : \ldots : a_K$, then $\sharp_c(i)$ is the least $k \leq K$ s.t. $p \leq \Sigma^{\mathrm{T}_c(i)-1}_{j=1}|Z^j| + \Sigma^k_{j=1} a_j$.

A cell c in $\mathcal{T}$ can be labeled $\overline{\alpha}^W$ iff coarity of c is equal to the coarity of W.

The *type* and *number* of a port i of c is defined as the type and number of i in W.

This technical definition is a complicated way of saying that given a port p of an encoding of a cell labeled α, p belongs to the splitting of one of α's principal ports. That is p's *type* (thus the special type I if unsplit). p is also one representative of a rule number for that port. That is its *number*.

Let us consider a cell c of a target system, labeled $\overline{\alpha}^{1:2,2:1,I,K}$. This means that the original system had at most 2-rules ($K = 2$, *i.e.* each type in the ID has size 2), the original cell has coarity 4 (there are 4 types in the ID) and that c has coarity $1 + 2 + 2 + 1 + 1 + K = 9$. The types and numbers of its principal ports is given in

$\mathrm{T}_c(1) = 1$	$\sharp_c(1) = 1$	First rule for first port
$\mathrm{T}_c(2) = \mathrm{T}_c(3) = 1$	$\sharp_c(2) = \sharp_c(3) = 2$	Second rule for first port
$\mathrm{T}_c(4) = \mathrm{T}_c(5) = 2$	$\sharp_c(4) = \sharp_c(5) = 1$	First rule for second port
$\mathrm{T}_c(6) = 2$	$\sharp_c(6) = 2$	Second rule for second port
$\mathrm{T}_c(7) = 3$	$\sharp_c(1) = I$ (undefined)	Not yet split third port
$\mathrm{T}_c(8) = 4$	$\sharp_c(8) = 1$	First rule for fourth port
$\mathrm{T}_c(9) = 4$	$\sharp_c(9) = 2$	Second rule for fourth port

Table 3.1: The detailed description the ports of a cell whose IDis $1:2, 2:1, I, K$.

the table on next page, along with what each of them is a representative of from the point of view of α.

Given an active pair $\overline{\alpha}_i^s \bowtie \overline{\beta}_j^t$, where the i-th port of $\overline{\alpha}$ has kind k_1 and number n_1 and the j-th port of $\overline{\beta}$ has kind k_2 and number n_2, there is a rule for it if and only if $n_1 = n_2 = n$ and there are at least n rules for $\alpha_{k_1} \bowtie \beta_{k_2}$ in the original system. The RHS of the rule depends of course on both the number and the kinds involved and are *almost* the translation of the original RHS, except that we have to take into account the fact that some ports have been split and copied.

When the original cells have a unique principal port, their encodings can have them split and duplicated. But as soon as one active pair $c \bowtie c'$ is reduced in the encoding, all other copies of the principal ports of c, c' become useless, since in the original system, the two principal ports involved in the interaction "disappear". Hence, it is sufficient to send a cell on each of these copies that says: "hey, I'm useless here. So are you now." For this purpose we use κ-cells. They have one port, which is principal. Unlike Lafont's ϵ, they do not erase cells but the ports they are connected to.

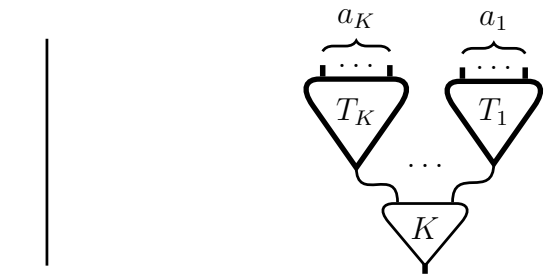
One the other hand, when the original cells have several principal ports that can be separately duplicated and split, the issue gets more complicated. If one copy interacts, all ports of the same type become useless, as in the former case. But ports of different type still represent ports that exists in the RHS of the rule, thus in the net resulting from the interaction. And these ports are already split and duplicated in some manner. If we just encode the RHS of the original rule as is, we get only unsplit and non-duplicated ports. This does not fit the interface of the left-hand side. We thus need to re-split and re-duplicate those translations of ports to end-up with the situation we had before the interaction. For this, we use a special kind of trees, we call for now *comtrees*. They have a root which is the principal port of a K-cell c. Each auxiliary port of c is connected to the root of a tree of χ-cells. The free ports of the tree (all auxiliary to some cell) are its leaves. The number of leaves is the arity of the tree.

Let us now consider a net μ of $\mathcal{S}$ which has an active pair $\alpha_i \bowtie \beta_j$ and its encoding ν. After reducing some active pair involving K and χ cells which belong to the communications zones linking the image of the ports i of the α-cell and j of β-cell, one obtains a net ν' which has an active pair $\overline{\alpha}_p^Z \bowtie \overline{\beta}_{p'}^{Z'}$ where $\mathrm{T}_Z(p) = i$, $\mathrm{T}_{Z'}(p') = j$

and $\sharp_Z(p) = \sharp_{Z'}(p') = k$ for some k.

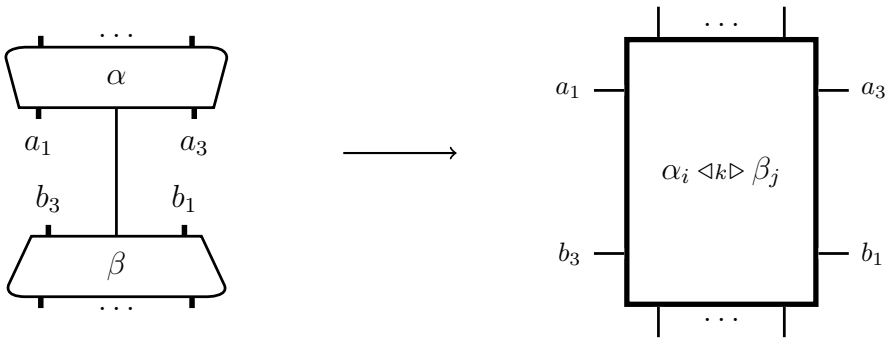
The RHS for $\alpha_i \bowtie \beta_j$ is the net $\alpha_i \lhd_k \rhd \beta_j$. Its image by the encoding is a net, in which each cell labeled ζ has been replaced by a cell labeled $\overline{\zeta}^{I,\ldots,I}$, with a unique copy of each principal port. Meanwhile, our active pair involves two cells where some of the active ports have been split and duplicated. For each type Y in Z and Z', we consider the comtree T:

- if $Y = I$, T is a simple wire;
- if $Y = a_1 : \ldots : a_K$, T is a tree of root cell c labeled K and such that the χ-tree attached to the k-th auxiliary port of it has arity a_k.

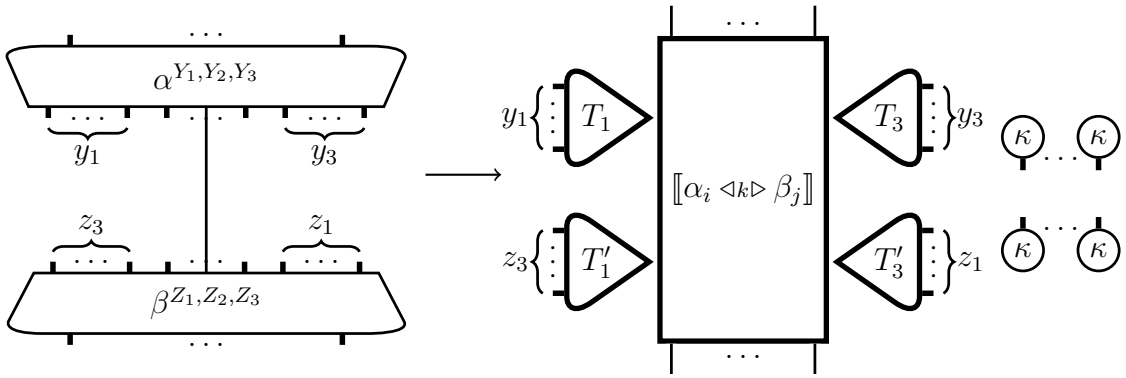


In the encoding, the image of a port in the interface of the active pair is now a certain amount of ports that can be determined by the corresponding type in Z or Z'. Depending on this type, we connect to the image of the RHS of the rule a comtree to the image of that exact port. Each ports of type i in $\overline{\alpha}$ and of type j in $\overline{\beta}$ receive a κ-cell.

Let us consider that α and β have same coarity 3 and that their encodings meet on ports of type 2 and number k. If the original k-th rule looked like this:



then an "image rule" lools like this:



where T_1 is comtree corresponding to Y_1, T_3 t Y_3, T_1' to Z_1 and T_3' to Z_3, while all ports of Y_2 and Z_2 receive κ-cells.

3.4.4 Encoding general nets into multiport nets

We now formalize all these ideas. Let $\mathcal{S} = (\Sigma_\mathcal{S}, \bowtie_\mathcal{S}, \mathcal{O}_\mathcal{S})$ be an arbitrary interaction net system. The only restriction we put is that all self-rules are symmetric. The case of asymmetric self-rules will be dealt with in a following section. We will define a simply wired INS $\mathcal{S}^*$ and an encoding $[\![\cdot]\!]$ of $\mathcal{S}$ into $\mathcal{S}^*$ such that, for every net μ of $\mathcal{S}$, $[\![\mu]\!]$ is simply wired.

Let K be the maximal number of rules for all active pairs and $\mathcal{I}d$ the set of identifications of K-types (see Definition 3.4.5). For each $n \in \mathbb{N}$, we denote by $\mathcal{I}d^n$ the set of identifications of $\mathcal{I}d$ which contain exactly n types (*i.e.* are composed of n K-types). Given an ID $Z = a_1^1 : \ldots : a_K^1, \ldots, a_1^n : \ldots : a_K^n$ of n K-types, we denote by $Z_{i(k)}$ the k-th element of its i-th type, in other words a_k^i. If the type is yet unsplit, then the notation is undefined. In every case, the notation Z_i represents the i-th type of Z.

Agents. The alphabet of $\mathcal{S}^*$ is composed of three symbols K, χ and κ, of degree K, 3 and 1, resp., and of a family of symbols $(\alpha^Z)_{Z \in \mathcal{I}d^n}$ for each symbol $\alpha \in |\Sigma_\mathcal{S}|$ of degree n, the degree of α^Z being the degree of Z. If $K = 1$ (simple rules), the K-cell is replaced by a simple wire. Intuitively, a cell of the form $\alpha^Z(\tilde{x}^1, \ldots, \tilde{x}^n)$ will represent an α cell whose principal ports are attached to connectors and might have been split and duplicated. In other words, we "embed" the connectors and the multirule choice in the cells themselves, modifying their degree if necessary.

K splits ports into K copies, each of same type as the original but of different number. It can only do so for unsplit ports. If $\sharp_Z(i) = I$,

$$\alpha^Z(x_1, \ldots, x_p, \ldots, x_n) \mid K(x_p; a_1, \ldots, a_K) \to \alpha^{\overline{Z}}(x_1, \ldots, x_{p-1}, a_1, \ldots, a_K, x_{p+1}, \ldots, x_n)$$

where $\overline{Z}$ is obtained from Z by replacing Z_i by $1 : \ldots : 1$ (written K).

However, where K is a constant of $\mathcal{S}$, the size of connectors changes dynamically (it is even altered by structural equivalence), so there is no hope of using a fixed α^Z

cell to encode a given α cell. The symbols χ and κ have precisely the purpose of ensuring that the "multiplicities" of ports are updated during reduction. They both have 1 principal port and interact with α^Z cells as follows. Let p be a port of α^Z of type i and number k.

$$\alpha^Z(x_1,\dots,x_p,\dots,x_n) \mid \chi(x_p;y,z) \to \alpha^{Z^+}(x_1,\dots,x_{p-1},y,z,x_{p+1},\dots,x_n)$$

where Z^+ is the ID obtained from Z by adding 1 to $Z_{i(k)}$ (a copy was made of a port of type i and rule k). If $Z_{i(k)} \geq 2$,

$$\alpha^Z(x_1,\dots,x_p,\dots,x_n) \mid \kappa(x_p) \to \alpha^{Z^-}(x_1,\dots,x_{p-1},x_{p+1},\dots,x_n)$$

where Z^- is the ID obtained from Z by subtracting 1 to $Z_{i(k)}$ (a port of type i and rule k has been deleted as it has become useless. At least one copy of such a port has to remain, thus the condition).

Connectors. To encode connectors, we use an idea of Ehrhard and Laurent [16] updated to include multirule translation.

Definition 3.4.6 (χ-tree, χ-zone, comnet)**.** A *χ-tree of root x* is a net defined inductively as follows: $[x,z]$ is a χ-tree of root x and arity 1; $\kappa(x)$ is a χ-tree of root x and arity 0; if X_1, X_2 are χ-trees of root a_1, a_2, then $\chi(x;a_1,a_2) \mid X_1 \mid X_2$ is a χ-tree of root x. The free ports of a χ-tree other than its root are called *leaves*; their number is the *arity* of the tree.

A *χ-zone* of order $n \geq 1$ on ports $x_0,\dots,x_{n-1}$ is a net structurally congruent to a net of the form

$$C_0(x_0;\tilde{a}^0) \mid \dots \mid C_{n-1}(x_{n-1};\tilde{a}^{n-1}) \mid \prod_{k=0}^{n-1}\prod_{i=1}^{n-1}[a_i^k, a_{n-i}^{(k+i) \bmod n}]$$

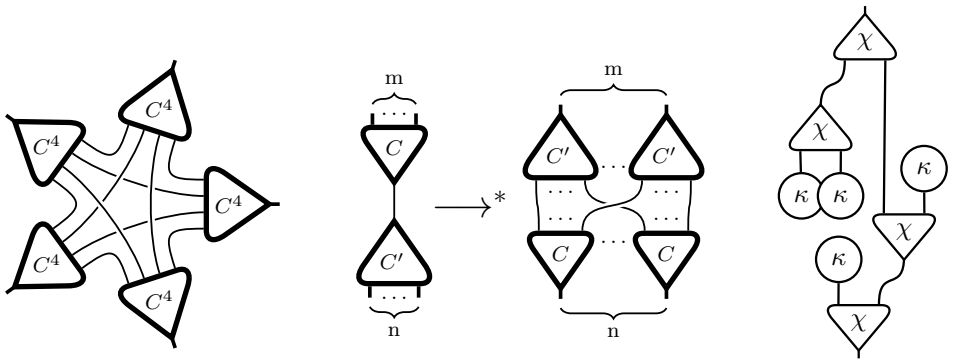
where, for all $0 \leq k \leq n-1$, $C_k(x_k;\tilde{a}^k)$ is a χ-tree of root x_k, arity $n-1$ and leaves $\tilde{a}^k$, and the sequences a^k are pairwise disjoint. We write $\texttt{Zone}(x_0,\dots,x_{n-1})$ to denote a generic χ-zone on $x_0,\dots,x_{n-1}$. The only χ-zone of order zero is the empty net.

A *communication net* (or *comnet*) of order $n \geq 1$ on ports $x_1,\dots,x_n$ is a net structurally congruent to a net of the form

$$K(x_1;\tilde{a}^1) \mid \dots \mid K(x_n;\tilde{a}^n) \mid \texttt{Zone}(a_1^1,\dots,a_1^n) \mid \dots \mid \texttt{Zone}(a_K^1,\dots,a_K^n)$$

where the sequences a^k have K elements and are pairwise disjoint. We use the notation $\texttt{Comm}(x_1,\dots,x_n)$ to denote a generic comnet on $x_1,\dots,x_n$. Again, the only comnet of order zero is the empty net.

Notice that the size of a comnet does not really depend on its arity. Consider a full tree T of χ-cells of height m (by full we informally mean that every auxiliary port of



(a) A χ-zone of order 5 (b) Reduction of χ-trees (c) χ-zone of order 2

Figure 3.8: Some examples of nets and reductions in $\mathcal{S}^*$.

a χ-cell that is not a leave of T is connected to the principal port of a χ-cell). It has 2^m leaves (and $2^m - 1$ cells, so it can get pretty big fast). Connect a κ-cell to $2^m - 1$ of them. You get a χ-tree of arity 1. Use that to build a comnet of order 2 (two of those and join their respective leaves together). Of course, when we are in control, we will use χ-trees and comtrees with no κ-cells. We call those *light trees*. For instance, the unique light χ-tree of order 1 is the simple wire.

Some particular cases can ease-up a little bit the readers indigestion. If $K = 1$, then the K-cell is not a cell but a wire, and a communication zone is a single χ-zone. We in fact fall into the MPW to MP encoding. Seemingly, if the original system is simply wired, every χ-zone needs to be at most of order 1, so simple wires are good enough. No need for χ-cells, and we get a quite natural MPR to MP encoding. In both cases, starting with a system with simple cells leads to the encodings discussed previously.

The behavior of comnets is governed by the following rules:

$$\begin{aligned}
\kappa(a) \mid \kappa(a) &\to 0,\\
\kappa(a) \mid \chi(a; y, z) &\to \kappa(x) \mid \kappa(y),\\
\kappa(a) \mid K(a; x_1, \ldots, x_K) &\to \kappa(x_1) \mid \ldots \mid \kappa(x_K),\\
\chi(a; w, x) \mid \chi(a; y, z) &\to \chi(w; a, b) \mid \chi(x; c, d) \mid \chi(y; a, c) \mid \chi(z; b, d),\\
\chi(a; w, x) \mid K(a; x_1, \ldots, x_K) &\to K(w; z_1, \ldots, z_K) \mid K(x; z'_1, \ldots, z'_K)\\
&\qquad \mid \chi(x_1; z_1, z'_1) \mid \ldots \mid \chi(x_K; z_K, z'_K)\\
K(a; x_1, \ldots, x_K) \mid K(a; y_1, \ldots, y_K) &\to [x_1, y_1] \mid \ldots \mid [x_K, y_K].
\end{aligned}$$

The 9 reduction steps introduced so far are called *administrative*, as are K,χ and κ cells, and purely administrative reductions are denoted by $\to_{\mathrm{a}}^*$. The following result is a tedious but straightforward verification. The second part uses the first.

Lemma 3.4.7.

1. *χ-trees reduce through $\to_{\mathrm{a}}^*$ as in 3.8(b).*

2. *For all symbols α^Y, β^Z of $\mathcal{S}^*$ of degree at least 1 and for all $n \in \mathbb{N}$, we have that*

$$\alpha^Y(\dots a_0 \dots) \mid \prod_{i=1}^{n} \mathtt{Comm}(a_{i-1}, a_i \dots) \mid \beta^Z(\dots a_n \dots) \to_{\mathrm{a}}^* \quad \alpha^{Y'}(\dots a \dots) \mid \beta^{Z'}(\dots a \dots) \mid \nu$$

for some net ν and some identifications Y', Z'.

It is sufficient for a *path that goes through comzones* to exist between two principal ports to be able to create an active pair between the two in the future. And this, unregarding of the geometry of the comzones: they might even be divergent (in the sense of paragraph 3.4.2).

The encoding, and its validity. We are ready to define the encoding:

$$\begin{array}{rlrl} [\![\alpha(\tilde{x})]\!] & = \alpha^{I,\dots,I}(\tilde{x}), & [\![0]\!] & = 0, \\ [\![[\tilde{x}]]\!] & = \mathtt{Comm}(\tilde{x}), & [\![\mu \mid \nu]\!] & = [\![\mu]\!] \mid [\![\nu]\!]. \end{array}$$

Since there are infinitely many comnets of any given order, the encoding is actually a relation. For simplicity, we will abusively denote by $[\![\mu]\!]$ any net in the image of μ through such a relation. Indeed, any of them faithfully encodes μ. It is obvious that $\mathrm{fp}([\![\mu]\!]) = \mathrm{fp}(\mu)$ and that $[\![\mu]\!]$ is simply wired (because comnets are simply wired).

As we can see, the image of a cell has a unique copy of each of the cells ports. Because of administrative steps, this number can later vary. For any K-type Z, we need to be able to transform a port of type I into one of type Z. For this, we define another special kind of tree.

Definition 3.4.8 (type-tree)**.** A *type-tree* of root x is a net structurally congruent to a net of the form

$$K(a, x_1, \dots, x_K) \mid X_1(x_1; \tilde{z}^1) \mid \dots \mid X_K(x_K, \tilde{z}^K)$$

where for any $1 \leq k \leq K$, X_k is a χ-tree.

It is a *type-tree for Z* if $Z = a_1 : \dots : a_K$ and for each $1 \leq k \leq K$, X_k has arity a_k. The *type tree for I* is a simple wire.

We may now go on with the definition of $\mathcal{S}^*$ and give its non-administrative interaction rules. Let $\alpha, \beta \in |\Sigma_{\mathcal{S}}|$ or respective arities m, n be s.t. $\alpha_i \triangleleft_k \triangleright \beta_j$ is defined. We denote by $\kappa(\tilde{a})$ the net containing all $\kappa(a_i)$, $a_i \in \tilde{a}$. For any $Y = Y_1, \dots, Y_m \in \mathcal{I}d^m$ and $Z = Z_1, \dots, Z_n \in \mathcal{I}d^n$, for any p, q such that $\mathrm{T}_Y(p) = i$, $\mathrm{T}_Z(q) = j$ and

$\sharp_Y(p) = \sharp_Z(q) = k$ we set

$$\alpha^Y(x_1, \ldots, x_p, \ldots, x_m) \mid \beta^Z(y_1, \ldots, y_q, \ldots, y_n) \to$$
$$[\![\alpha_i \lhd_k \rhd \beta_j]\!]\{\tilde{a}/\tilde{p}, \tilde{b}/\tilde{q}\} \mid \prod_{l \neq i} T^{Y_l}(a_l; \tilde{x}^l) \mid \prod_{l \neq j} T^{Z_l}(b_l; \tilde{y}^l) \mid \kappa(\tilde{a}^i) \mid \kappa(\tilde{b}^{j'}),$$

whenever $x_p = y_q$, where T^{Y_l}, T^{Z_l} are type-trees for l-th type of Y and Z resp. The intuition is that as soon as a "copy" of port i of (the encoding of) an α cell is connected to a "copy" of port j of a β cell, the interaction $\alpha_i\beta_j$ is simulated through the encoding, and administrative cells are introduced to perform the necessary bookkeeping. In particular, κ cells are dispatched to all the remaining "copies" of port i of α and j of β, which are no longer necessary. Of course, to actually define the rule, we need to fix a representative of $[\![\alpha_i \lhd_k \rhd \beta_j]\!]$ and the comtrees T^{Y_l}, T^{Z_l}, the actual choice being irrelevant. We can still add the condition that the choices we make for comzones and χ-trees all use light trees. It is a simple complexity choice, unnecessary, but it ensures at least one nice result:

Proposition 3.4.9 (Useless theorem)**.** *If $\mathcal{S}$ is a* MP *system (i.e. with simple wires and simple rules) and the encoding from $\mathcal{S}$ to $\mathcal{S}^*$ only uses light trees, then $\mathcal{S}^* = \mathcal{S}$ where (useless) $I, \ldots, I$ identifications have been added to all labels.*

Proof. The K-cell being in fact a simple wire, as is the only light χ-tree of order one, each wire (which is simple) is translated into a list of two-by-two connected simple wires which is structurally congruent to a simple wire. No K nor χ-cells, so each $\alpha^{I,\ldots,I}$ cells stays as is is until real interaction, so no other ID is ever needed. It is sufficient to relabel them α. □

The name of the proposition is justified by the fact that we are not interested at all by encoding a system into itself. It is nice anyhow to stress that our encoding is so faithful that is does not modify what need not be.

The specification of the MP system $\mathcal{S}^*$ is completed by declaring that a non-administrative reduction is observable as soon as the corresponding reduction of $\mathcal{S}$ is. The reader may check that no reduction rule of $\mathcal{S}^*$ introduces connectors of order other than 2, so $\mathcal{S}^*$ is simply wired.

One of the main tools we use to show equivalences is the fundamental determinacy of some of its cells and interactions, namely administrative ones. Determinacy is here understood as some local confluence.

As we have mentioned, all χ-trees of same arity are congruent. We have not proved so yet, but we need to express that two nets only differ by some χ-trees.

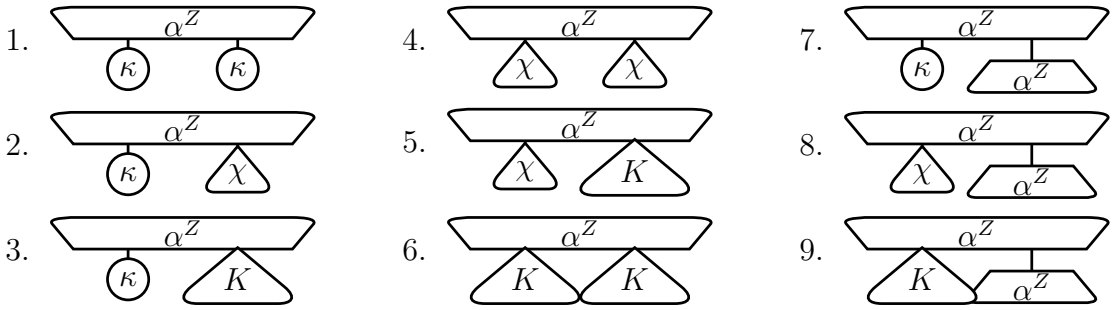
Definition 3.4.10. Let $\sim_{\mathrm{T}}$ be the smallest congruence on nets containing structural congruence and such that $T \sim_{\mathrm{T}} T'$ for all χ-trees T, T' of equal arity with same root and leaves.

For instance, for any type Y, all type-trees for Y are $\sim_{\mathrm{T}}$.

Lemma 3.4.11 (Determinacy of administration)**.** *Let μ be a net of $\mathcal{S}^*$ s.t. $\mu \to_{\mathrm{a}} \mu_1$ and $\mu \to \mu_2$. Then there are two reductions $\mu_1 \to^* \nu_1$ and $\mu_2 \to^* \nu_2$ such that $\nu_1 \sim_{\mathrm{T}} \nu_2$. Moreover,*

- *if both initial interactions are administrative, then $\nu_1 = \nu_2$, $\mu_1 \to^* \nu_1$ and $\mu_2 \to^* \nu_2$ have same length which is 0 or 1.*
- *if $\mu \to \mu_2$ is non-administrative and the two active pairs share a cell, then $\mu_1 \to^* \nu_1$ has length 1 and is non-administrative while $\mu_2 \to^* \nu_2$ has length 0.*

Proof. If the two reductions do not share cells, then the result is immediate. All administrative cells have one principal port, so for two active pairs to share a cell, that one has to be non-administrative. The "only" cases remained to study are the following:



Of course, not every ID Z fits for every reduction, even if the ID is of correct size. For instance, in case 1, for both reductions to be triggable, the type and number of each port connected to the κ-cells has to contain at least two copies. In almost each case, some special subcases occur. Again in 1, if both involved ports belong to the same type and number, then the two interactions already result in the same net $\mu_1 = \mu_2$ (it is actually the only situation of the first kind when the second reductions have length 0). In general, in any of the cases 1 through 6, the reductions can be considered almost independent: reducing one active pair maintains the other one, in which the non-administrative cell has been replaces by an $\alpha^{Z'}$ cell for some Z', on which a reduction can still be applied, bearing the desired result.

The important cases are the last three ones, that are all similar. Let us call c, d the non-administrative cell and a the administrative one connected to c. The administrative reduction transforms the ID of the cell c in exactly one type $Z_i \in Z$ which becomes Z_i'. The new cell c' is still connected to the other non-administrative cell; reducing it produces a type-tree for Z_i'. On the other hand, the non-administrative step reduces to a net in which a is connected to a leave of a type-tree for Z_i. If a is a κ or a χ-cell, then the type-tree for Z_i along with cell a form a type-tree for Z_i', not necessarily the one chose for the encoding, but $\sim_{\mathrm{T}}$, thus the result. If a is a K-cell, then, even better: μ_1 reduces directly to μ_2, since $Z_i = I$, $Z_i' = K$, so Z_i's type-tree is exactly a K-cell. □

To understand $\sim_{\mathrm{T}}$ a little better, we need to understand interactions between χ-trees and arbitrary cells.

Lemma 3.4.12. *Let c be any cell and p one of its principal ports, and let T, T' be any two χ-trees of arity k. Let us consider full reductions of $c \mid T(p, \tilde{u}) \to^* \mu$ and $c \mid T'(p, \tilde{u}) \to^* \nu$. Then $\mu \sim_{\mathrm{T}} \nu$.*

Proof. In fact:

- If c is a κ-cell, then $c \mid T(p; \tilde{u}) \to^* \kappa(\tilde{u})$ and $c \mid T'(p; \tilde{u}) \to^* \kappa(\tilde{u})$;
- If c is a χ or K cell, and $c \mid T(p; \tilde{u})$ fully reduced to some net μ, then $c \mid T'(p; \tilde{u})$ reduces to the net ν, which is μ with each copy of T replaced by T' (so $\mu \sim_{\mathrm{T}} \nu$);
- If c is a non-administrative cell of label α^Z, then p is of a certain type t and of certain number n in Z (otherwise, if p is unsplit, the reduction cannot take place and the result is trivial, except if $K = 1$, which brings us back to the same case). Let l be the value of type t and number n in Z. Then $\mu = \nu = \alpha^{Z'}\{\tilde{u}/p\}$ where Z' is Z in which the value l has been replaced by $l + k$.

□

The only particularity might arise when the considered trees have arity 1, since then one of T, T' might be a simple wire, which can be a little awkward. It is because of that situation that $\sim_{\mathrm{T}}$ is not yet a barbed bisimulation.

Lemma 3.4.13. *Let $\sim_{\mathrm{a}}$ be the smallest congruence on nets containing $\sim_{\mathrm{T}}$ and such that $\mu \sim_{\mathrm{a}} \mu'$ for all $\mu \to^*_{\mathrm{a}} \mu'$.*

Then $\sim_{\mathrm{a}}$ is a barbed bisimulation. Moreover, $\mu \sim_{\mathrm{a}} \nu$ and $\mu \to^* \mu'$ by a reduction containing n non-administrative steps implies $\nu \to^* \nu' \sim_{\mathrm{a}} \mu'$ also by a reduction containing n non-administrative steps.

Proof. Note that, by symmetry of $\sim_{\mathrm{a}}$, it is enough to show that it is a simulation. The key observation is that $\mu \sim_{\mathrm{a}} \nu$ implies that there is a one-to-one correspondence between non-administrative cells of μ and ν, and that a non-administrative (α^Y, β^Z)-active pair in μ yields in ν the presence of two cells $\alpha^{Y'}$ and $\beta^{Z'}$ connected by a path composed of χ and K-cells, which may be reduced to perform the simulation. □

Note that Lemma 3.4.13 implies that $\mu \simeq^{\mathrm{c}}_{\mathcal{S}^*} \nu$ as soon as $\mu \sim_{\mathrm{a}} \nu$, which is quite useful.

Theorem 3.4.14. *For every net μ of $\mathcal{S}$, we have:*

- *$\mu \to^* \mu'$ implies $[\![\mu]\!] \to^* \simeq^{\mathrm{c}}_{\mathcal{S}^*} [\![\mu']\!]$;*
- *$[\![\mu]\!] \to^* \nu$ implies that there exists μ' s.t. $\mu \to^* \mu'$ and $\nu \to^* \simeq^{\mathrm{c}}_{\mathcal{S}^*} [\![\mu']\!]$.*

Proof. Completeness is a consequence of 3.4.7 and of the fact that $\mu \equiv \mu'$ in $\mathcal{S}$ implies $[\![\mu]\!] \simeq^{\mathrm{c}}_{\mathcal{S}^*} [\![\mu']\!]$, which follows immediately from Lemma 3.4.13. For what concerns soundness, we prove it by induction on the number of non-administrative steps in the reduction $[\![\mu]\!] \to^* \nu$, using the second part of Lemma 3.4.13 in a similar way to how we used 3.2.3 in the proof of soundness for 3.2.5. □

Theorem 3.4.15. *For every net μ of $\mathcal{S}$, $\mu \;{}_{\mathcal{S}}\dot{\approx}_{\mathcal{S}^*} [\![\mu]\!]$.*

Proof. The relation $\{(\mu, \nu) \in \mathcal{S} \times \mathcal{S}^* \;;\; \nu \sim_{\mathrm{a}} [\![\mu]\!]\}$ is an $(\mathcal{S}, \mathcal{S}^*)$-barbed bisimulation. In complement to the homomorphism of the encoding, it proves the theorem. □

Combined with this theorem, lemma 3.4.13 says that the simulation of a reduction in $\mathcal{S}^*$ has as many non-administrative step as the original reductions had steps in general. Therefore, divergence can only occur from administrative interactions. Considering a size of a net to be a pair composed of, first the quantity of χ and K-cells in a net, and second the quantity of κ-cells, we can see the only administrative rule that does not strictly decrease the size is the rule for $\chi \bowtie \chi$. And in fact, it is possible to build a simple net composed of χ-cells that diverges using this rule only (see p. 117). In encodings, such a net occurs, and it is only the case when two communication zones share more than one port (or a communication zone is degenerate $\mathtt{Comm}(a, a, \tilde{b})$.

Theorem 3.4.16 (Divergence of the encoding)**.** *For any net $\mu \in \mathcal{S}$, $[\![\mu]\!]$ has an infinite reduction iff one at least of the following holds:*

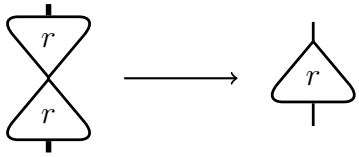
- *μ has an infinite reduction;*
- *$\mu \to^* \nu$ and ν has a vicious connector.*

In fact, if it wasn't for this divergent encoding of wires that share more than one name, we could consider a structural congruence equation for wires of the form $[\tilde{a}] \mid [\tilde{b}] \equiv [\tilde{a} \cup \tilde{b}]$ as soon as $\tilde{a} \cap \tilde{b} \neq \emptyset$, slightly more elegant. Anyhow, the refinement of simple wires is only made stronger by this: they are powerful, and sensitive to short-circuits.

In short, the encoding does not introduce divergence: $[\![\mu]\!]$ diverges only if μ diverges in the sense of 3.1.8, *i.e.* it generates an infinite reduction or a vicious connector. We understand it is debatable that the presence of a vicious connector should be considered as divergence. The fact alone of controlling when the encoding diverges, and that is does so only for very administrative reasons is, we believe, an argument in favor of this encoding.

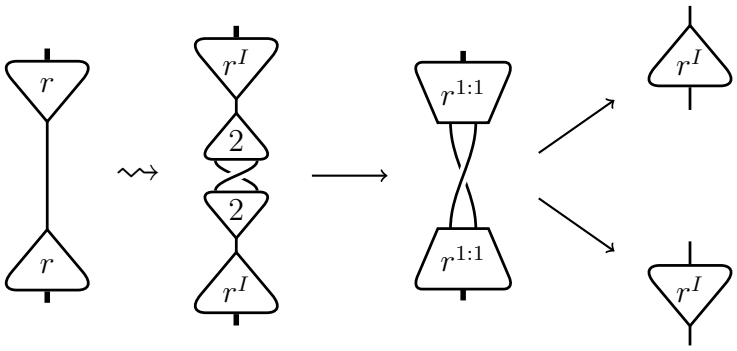
3.5 Multiwires can express rule ambiguity

In our encoding of general interaction nets into MP, we did not take into account asymmetric reflexive rules as they have a very particular way of bringing ambiguity. Consider the simple example below composed of a unique cell r with arity and coarity 1 and a single non-symmetric rule:



$$r(a;x) \mid r(a;y) \to r(x;y)$$

It encompasses its symmetric result $r(a;y) \mid r(a;x) \to r(y;x)$. If simply considered as two different rules, its encoding in MP would go like this:



The last two interactions shown in the picture represent the two *versions* of the asymmetric reflexive rule. But as pure multiport rules, they are each non-symmetric as well, and there is no way to actually define the two rules to make sure the two issues come out. As asymmetric rules, they could both lead to an r-cell pointing north.

We did not insist on that problem in the previous section, since it complicates an already complicated encoding. Also, we deal very specifically with it in this section that is divided in three parts. First, we show that the only multirule that is necessary for expressing all multirules is in fact the one shown above. It has a non reflexive version that can be used in case the original multirule system has no self-rule and one wishes to keep this property by encoding. This way, we mostly get rid of multirules. Then we use a translation of this special asymmetric rule to encode MR into MW (we let the reader imagine how the versions with no-self rules can be applied). The extension of that encoding to an encoding from MWR to MW is then straightforward. Finally we discuss how to use this to really encode multirules into multiports.

3.5.1 One multirule to rule them all

The first result has already been noticed by Alexiev ([1]), in a weaker form. He shows that it is possible to get rid of multirules that are not reflexive. For this, he encodes a wire into a net that randomly choses a $k \in \{1, \ldots, K\}$ that decides in some sense which rule should be used. This net uses deterministic rules and one asymmetric reflexive one. A little problem of this net is that it is not symmetric itself, so Alexiev settle for an ordering of labels in order for the encoding to decide in which direction to encode

each wire, a little like comzones. Since it is possible to avoid this inconvenience, let us show how.

We take this observation of his to a next level. It is possible to get rid of all multirules, even asymmetric reflexive ones, using a unique asymmetric reflexive rule on a simple cell, shown in introduction of the section. Even better, if the original system has no self-rules, it is possible to do the same trick, using a unique non-self-pair that has two rules, all other rules being simple.

In presence of reflexive rules

Let r be a cell of arity and coarity 1, with its reflexive rules

$$r(a;x) \mid r(a;y) \to r(x;y)$$

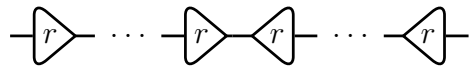
Theorem 3.5.1 (Multirule systems are *almost* Lafont-nets)**.** *Let $\mathcal{S}$ be a multirule system. Then there is another multirule system $\mathcal{S}^*$ that contains the cell r, s.t. $\mathcal{S}^*$ is a valid encoding of $\mathcal{S}$ and the unique multirule of $\mathcal{S}^*$ is the asymmetric self-rule for r.*

Proof. Basically, we build, for every $K \in \mathbb{N}$, a net K with two free ports x, y, that reduces, for any $1 \leq k \leq K$, to a net R^k *pointing towards* x or y. So the net K by itself has $2K$ normal forms.

The nets R^k are composed of a chain of r cells, connected back to front:

- R^0 is a wire;
- $R^1(a,b) \equiv r(a,b)$;
- $R^k(a,b) \equiv R^{k-1}(a,x) \mid r(x,b)$.

Fact (a)**.** *$K \equiv R^K(a,x) \mid R^K(a,y)$ can reduce to $R^k(x,y)$ and $R^k(y,x)$ for any $1 \leq k \leq K$. Moreover, during the reduction, before it has reached one of these (normal) forms, both its free ports are auxiliary ports of an r cell, and therefore cannot interact with the environment.*

$K \equiv$ 

Proof. For any $k, l \leq 1$, $R^k(a,x) \mid R^l(a,y)$ can reduce in one step to one of $R^{k-1}(a,x) \mid R^l(a,y)$ or $R^k(a,x) \mid R^{l-1}(a,y)$, both with auxiliary ports in the interface except if one of l or k equals 1, in which case one of the normal forms has been reached. □

For every cell label α in $\mathcal{S}$, there is in $\mathcal{S}^*$ a family of cells $\{\alpha^i\}_{0 \leq i \leq K}$ with same arity as α. They behave like counters:

$$\forall 0 \leq k \leq K-1, \quad \alpha^k(a;\tilde{z}) \mid r(a;b) \to \alpha^{k+1}(b;\tilde{z}).$$

Fact (b)**.** *$\alpha^0(a;\tilde{z}) \mid K(a,b) \mid \beta^0(b,\tilde{z}')$ reduces either to $\alpha^k(c;\tilde{z}) \mid \beta^0(c;\tilde{z}')$ or to $\alpha^0(c;\tilde{z}) \mid \beta^k(c;\tilde{z}')$ for some $k \in \{1,\dots,K\}$.*

What is an asymmetric reflexive rule? It is rule on a net of the form $\alpha(a,\tilde{z}) \mid \alpha(a,\tilde{z}')$ such that the result is not symmetric w.r.t. $\tilde{z},\tilde{z}'$:

$$\alpha \lhd\rhd \alpha\{\tilde{z}/\tilde{z}'\} \not\equiv \alpha \lhd\rhd \alpha.$$

This means it is possible to consider each of those as *oriented towards* $\tilde{z}$ or towards $\tilde{z}'$. So if we in some sense manage to transform each active pair $\alpha \bowtie \alpha$ into an active pair $\alpha \bowtie \alpha'$, transforming one only of the two cells labeled α into a cell labeled α', and consider the rule for $\alpha \bowtie \alpha'$ as the rule for $\alpha \bowtie \alpha$ oriented toward the "second" cell α, we transposed the asymmetry from the rule to a pre-choice. We denote the choice "towards the first cell" by $\overleftarrow{\alpha \lhd\rhd \alpha'}$ and the other choice $\overrightarrow{\alpha \lhd\rhd \alpha'}$. For symmetric self-rules or non reflexive rules, we can consider both *oriented* rules to be the same. We have seen in Fact (b) that we are able to "asymmetricize" a reflexive active pair, in a non deterministic way, in the case $\alpha = \beta$.

Whatever the encoding will be we can already give the translation of the oriented versions of rules:

$$\alpha^k(a,\tilde{y}) \mid \beta^0(a,\tilde{z}) \to [\![\overleftarrow{\alpha \lhd_k \rhd \beta}]\!]$$

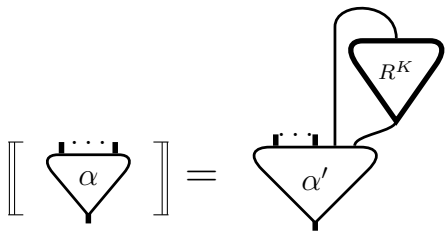
if there is a rule k for $\alpha \bowtie \beta$ in $\mathcal{S}$, and

$$\alpha^k(a,\tilde{y}) \mid \beta^0(a,\tilde{z}) \to [\![\overleftarrow{\alpha \lhd_K \rhd \beta}]\!]$$

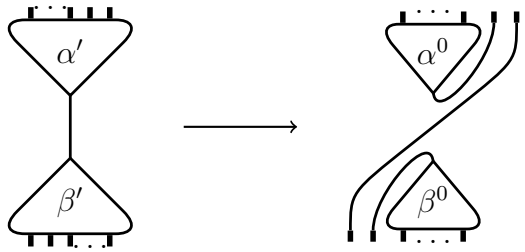
otherwise (any predetermined choice would do). It is observable in $\mathcal{S}^*$ iff the k-th rule for $\alpha \bowtie \beta$ is observable in $\mathcal{S}$ (or the K-th in the second case).

For the encoding, we could just transform each cell α into a cell α^0 preceded on its principal port by a R^K net. But such a solution could start a choice of rule process between two cells that have no rule at all.

Instead, the encoding of a cell α is composed of a cell α' of arity $\mathtt{ar}(\alpha)+2$, and a net R^K:



If $\alpha \bowtie \beta$ has at least one rule in $\mathcal{S}$, the rule for $\alpha' \bowtie \beta'$ is:



In this way, if we are sure α, β can interact, we fall into the situation we wanted. And only in that case. Notice also that this rule is simple, even when $\alpha = \beta$.

Lemma 3.5.2. *This encoding is valid.*

> *Proof.* Compositionality is completely met. Name invariance is undoubtful. Operational correspondence is given by Fact (b). Divergence reflexion is not in question. Barbs are the same for a net and its encoding. □

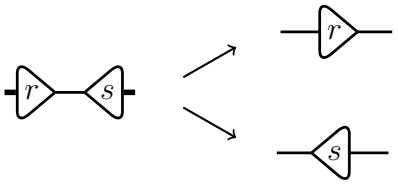
This concludes the proof of the theorem.

□

If the rules are not reflexive

The exact same strategy can be applied avoiding self-rules by differentiating two kinds of r-cells, in the way Lafont creates oriented combinators [33]. Let us consider two cells r and s of arity and coarity 1. The two rules that govern them are:

$$r(a;x) \mid s(a;y) \to r(y;x)$$
$$r(a;x) \mid s(a;y) \to s(x;y)$$

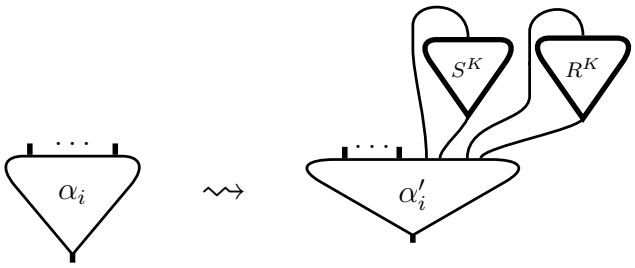


We now consider the nets R^k and S^k to be composed of chains of r and s cells respectively, connected back to front:

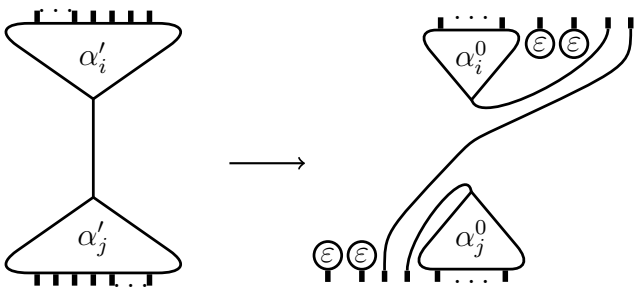
- R^0 is a wire;
- $R^1(a,b) \equiv r(a,b)$;
- $R^k(a,b) \equiv R^{k-1}(a,x) \mid r(x,b)$.
- S^0 is a wire;
- $S^1(a,b) \equiv s(a,b)$;
- $S^k(a,b) \equiv S^{k-1}(a,x) \mid s(x,b)$.

Fact (a'). *$K \equiv R^K(a,x) \mid S^K(a,y)$ can reduce to $R^k(y,x)$ and $S^k(x,y)$ for any $1 \leq k \leq K$. Moreover, during the reduction, before it has reached one of these (normal) forms, both its free ports are auxiliary ports, and therefore cannot interact with the environment.*

Let us now consider a multirule system $\mathcal{S}$ with no self-rules. We can order its alphabet without any loss of generality, so we can consider the labels to be $\alpha_1, \ldots, \alpha_n$. The translation of a cell α_i is given by a cell $\overline{\alpha}_i$ or arity $\mathtt{ar}(\alpha_i) + 4$ and two nets, R^K and S^K:



Suppose now $i < j$, then the interaction of α_i with α_j will pick for the first the net R^K and second the net S^K, sending some erasors on the ports of the unused net:



3.5.2 Encoding the asymmetric rule using multiwires

We will use this result on multirules for our translation from MWR to MW. We encode the $r \bowtie r$ rule of the previous section by a symmetric rule that uses multiwires. It is not complicated, and it reveals an interesting (and fashionable) point of view on non-determinism in calculation[2].

How can we brake the symmetry of an encoding of $r(a, x) \mid r(a, y)$ to favor either x or y? We do not really. What we do instead is that we *superpose* the two possible outcomes in one net, and let the environment decide.

Let us consider a cell qb of arity and coarity 1 in a system with multiwires. The only reflexive rule for qb is

$$qb(a; x) \mid qb(a; y) \rightarrow qb(b; c) \mid [b, c, x, y]$$

Let $\mathcal{S}$ be a source language which is MR and $\mathcal{T}$ the target language which is MW. Let K be the maximum number of rules for an active pair, considering that the two versions of an asymmetric self-rule count as one.

[2]We will not pronounce *The Word*, as we do not develop in any sense that metaphor in this work and would not like an automatic referencing to suggest that we do.

Along with the new cell qb, for every $\alpha \in \mathcal{S}$, there is a $\alpha' \in \mathcal{T}$ which has arity $\mathtt{ar}(\alpha)+2$ and a family $\alpha^0, \dots, \alpha^K$ which have the same arity as α.

Again, we define some special nets. For every $k \in \mathbb{N}$,

$$R^k(a,x) \equiv qb(a;x_1) \mid qb(x_1;x_2) \dots \mid qb(x_{k-1};x)$$

is a chain of qb-cells (meaning a wire if $k=0$), and, for every $k \in \mathbb{N}^*$

$$\overline{R}^k(a,x) \equiv [a,y,z,x_1] \mid qb(y,z) \mid qb(x_1,x_2) \dots \mid qb(x_{k-1},x)$$

is a chain of qb-cells with a qb-cell connected by its both ports to the head of the chain ($S^1(a,x)$ is just a qb-cell with its auxiliary and principal port connected on a same connector).

Encoding a cell α looks exactly like the previous encoding:

$$[\![\alpha(a;\tilde{z})]\!] = \alpha'(a;s,t,\tilde{z}) \mid R^K(t,s)$$

Wires stay wires.

The *administrative* rules also look alike:

$$\begin{aligned}
\alpha'(a;x,x',\tilde{x}) \mid \beta'(a;y,y',\tilde{y}) &\to \alpha^0(x',\tilde{x}) \mid \beta^0(y',\tilde{y}) \mid [x,y] \qquad \text{if } \alpha \bowtie \beta \text{ has a rule in } \mathcal{S} \\
\alpha^k(a,\tilde{x}) \mid qb(a,b) &\to \alpha^{k+1}(b,\tilde{x}) \\
qb(a,x) \mid qb(a,y) &\to qb(b,c) \mid [b,c,x,y]
\end{aligned}$$

Lemma 3.5.3. *Let $K(a,b)$ be the symmetric net $R^K(x,a) \mid R^K(x,b)$ (with both ports of the interface auxiliary). Then, for any $1 \leq k \leq K$, $K(a,b)$ reduces to $\overline{R}^k(a,b)$ or to $\overline{R}^k(b,a)$. These are the only reductions of $K(a,b)$ for which one of the ports of the interface is principal (both iff $k=1$).*

Proof. The proof is by induction. We do not give it in detail. In the following, we denote by $Qb(a,a')$ a net structurally congruent to $[a,a',s,t] \mid qb(s,t)$. The first step of reduction of $K(a,b)$ leads to the nets

$$R^{K-1}(x,a) \mid Qb(x,y) \mid R^{K-1}(y,b)$$

which corresponds to the first step of the induction if $K=1$. Then, for every $k,l \leq K-1$, $R^k(x,a) \mid Qb(x,y) \mid R^l(y,b)$ has two possible reductions:

$$\begin{aligned}
R^{k-1}(x,a) \mid Qb(x,y) \mid R^l(y,b) \\
R^k(x,a) \mid Qb(x,y) \mid R^{l-1}(y,b)
\end{aligned}$$

which interfaces are auxiliary. So the reductions of the net are defined by pairs of (k,l) decreasing one coordinate at the time. Until one of k,l *becomes* 0, which is a

case given by the lemma. □

We can see that this looks like the simulation of asymmetric rules given in the previous section. The difference is that here, when $K(a,b)$ reaches one of the forms $S^k(a,b)$ or $S^k(b,a)$, it is not normal and can go on *decreasing* to $S^{k-1}(a,b)$ or $S^{k-1}(b,a)$ until it reaches $Qb(a,b)$ (in both cases). But all these cases fall in the conclusion of the lemma.

Lemma 3.5.4. *Let* $Act \equiv \alpha^0(a,\tilde{x}) \mid R^K(s,a) \mid R^K(t,b) \mid \beta^0(b,\tilde{y})$. *Then all possible reductions reach one of*

- $\alpha^k(a,\tilde{x}) \mid \beta^0(a,\tilde{y})$ *for some* $1 \leq k \leq K$;
- $\alpha^0(a,\tilde{x}) \mid \beta^k(a,\tilde{y})$ *for some* $1 \leq k \leq K$;

Proof. We proceed by pieces. Thanks to Lemma 3.5.3, we have that *Act* necessarily reduces to a net of the form, for some $1 \leq k \leq K$:

$$\alpha^0(a,\tilde{x}) \mid S^k(a,b) \mid \beta^0(b,\tilde{y})$$

or

$$\alpha^0(a,\tilde{x}) \mid S^k(b,a) \mid \beta^0(b,\tilde{y}).$$

Fact (1)**.** *Let* $P^k(b,\tilde{x}) \equiv \alpha^0(a;\tilde{x}) \mid S^k(a,b)$. *Then all possible reductions of* P *necessarily reach one of:*

- $L(b,\tilde{x}) \equiv \alpha^0(a;\tilde{x}) \mid Qb(a,b)$;
- $W^l(b,\tilde{x}) \equiv \alpha^l(b;\tilde{x})$ *for some* $l \leq k$.

These are the only cases when b *is principal.*

Proof. Two possibilities only for P^k:

- either reducing the active pair $\alpha^0 \bowtie qb$ leading to a net $\alpha^1(a;\tilde{x}) \mid R^{k-1}(a,b)$ which deterministically leads to $\alpha^k(b,\tilde{x}) = W^k(b,\tilde{x})$;
- or the only active pair $qb \bowtie qb$, which leads to $P^{k-1}(b,\tilde{x})$.

Reiterating this choice as long as the second one is chosen proves the fact, $P^1(b,\tilde{x})$ being exactly $L(b,\tilde{x})$. □

Fact (1) says that in a net $\alpha^0(a,\tilde{x}) \mid S^k(a,b) \mid \beta^0(b,\tilde{y})$, no interaction is possible on β^0 unless the left part of the net hasn't reached $\alpha^0(a,\tilde{x}) \mid Qb(a,b)$, which has two reductions:

$$\alpha^1(a,\tilde{x}) \mid \beta^0(a,\tilde{y}) \qquad \text{or} \qquad \alpha^0(a,\tilde{x}) \mid \beta^1(a,\tilde{y}).$$

This finishes the proof of the lemma. □

We can now give the last set of rules, which actually encode the rules of $\mathcal{S}$.
If $\alpha \neq \beta$:

$$\begin{array}{l} \alpha^k(a;\tilde{x}) \mid \beta^0(a;\tilde{y}) \\ \alpha^0(a;\tilde{x}) \mid \beta^k(a;\tilde{y}) \end{array} \quad \begin{array}{c} \searrow \\ \nearrow \end{array} \quad [\![\alpha \lhd_k \rhd \beta]\!]$$

Otherwise:

$$\begin{array}{l} \alpha^k(a;\tilde{x}) \mid \alpha^0(a;\tilde{y}) \to \overleftarrow{\alpha(a;\tilde{x}) \lhd_k \rhd \alpha(a;\tilde{y})} \\ \alpha^0(a;\tilde{x}) \mid \alpha^k(a;\tilde{y}) \to \overrightarrow{\alpha(a;\tilde{x}) \lhd_k \rhd \alpha(a;\tilde{y})} \end{array}$$

For exactly the same reasons as in the previous section,

Lemma 3.5.5. *This encoding is valid.*

Moreover, it we allow multiwires in the original system $\mathcal{S}$, the encoding is still valid. We indeed made sure that a simulation is triggered if at least a rule can be applied. So this translation actually gives us an encoding from MWR to MW, where connectors are encoded as connectors.

It is simple to apply the second result of Section 3.5.1 (p. 132) by using two cells r, s along with qb to get rid of non-reflexive rules with only non-reflexive rules. The only rule for $r \bowtie s$ yields a qb-cell which can both interact with r and s in a unique way, providing in this way a translation form a self-rule-free multirule system to a self-rule-free multiwire system.

Notice that if the original system has no multirule, then we have to consider $K = 0$ ($K = 1$ means all active pairs have one rule only, but some self-rule might be asymmetric).

3.5.3 What about multiports?

It is possible to encode the reflexive r-cell into a multiport cell and proceed with the exact same schema to encode multirules. We prefer to complete the encoding from general interaction nets to strictly multiport ones given in Section 3.4 by showing how to encode asymmetric self-rules in it. Interestingly, the solution gives us a hint on how to separate multiports from multiwires (see Section 3.7).

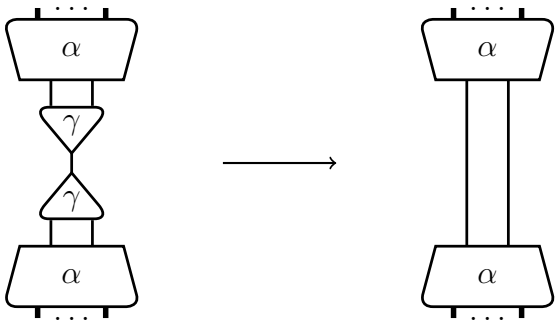
We need to brake the symmetry. As we have seen, our encoding duplicates ports but does not brake the symmetry. Let us consider a system that has a cell which has only one asymmetric reflexive rule, and the following encoding:

$$[\![\alpha(a;\tilde{x})]\!] = \alpha^2(a', a''; \tilde{x}) \mid \gamma(a; a', a'').$$

γ is nothing more that Lafont γ-cell:

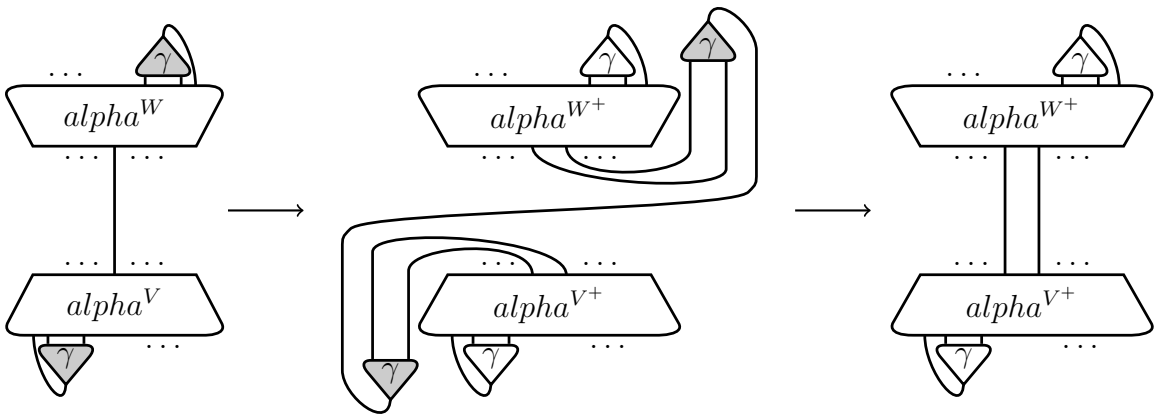
$$\gamma(a, x_1, x_2) \mid \gamma(a, y_1, y_2) \to [x_1, y_2] \mid [y_1, x_2].$$

This give us the following encoding for the active pair α/α:



which can reduce in two ways, depending on which active pair is considered. And in a completely deterministic way, since the active pairs are now not reflexive.

We now fit this in the multiport framework by transposing what we did with the multiwire translation. Multiport cells now have extra three auxiliary ports, on which we plug the γ-cell as shown in the following picture. We do not give the detail of the encoding, but just show how the simulation works, only in the case of simulating a symmetric rule: $\alpha_i \bowtie \alpha_j$ where $\mathrm{T}_V(i) = \mathrm{T}_W(j)$. Once the communication zone has created the active pair between the two ports of same type representing a same rule – the maybe asymmetric self-rule – the simulation continues like this:



A new γ-cell is created by reduction in case another symmetric active pair would want to get a try at being simulated. At the end, we get the asymmetric look we are looking for. Of course the IDs V^+ and W^+ have to reflect what those two new ports mean, so in general identifications have to be changed a bit, and all encoding cells should now come with this extra γ-cells. There is some technical work to be done to make sure these additions do not change the result.

3.6 Multirules alone do not give concurrency

In the following, we fix an arbitrary INS $\mathcal{S}$.

Definition 3.6.1 (Must observability)**.** A port x is said to be *must-observable* in the net μ if, for all μ' s.t. $\mu \to^* \mu'$, we have $\mu' \Downarrow_x$. In that case, we write $\mu \Downarrow\!\!\Downarrow_x$.

Observe that, by definition, must observability is preserved by reduction.

Lemma 3.6.2. *Let x be a port and let $\mu \equiv \mu' \mid \alpha(y; \tilde{z})$ be a net of $\mathcal{S}$, with $y \neq x$, $y \in \mathrm{fp}(\mu)$. Then, $\mu \Downarrow\!\!\Downarrow_x$ iff $\mu' \Downarrow\!\!\Downarrow_x$.*

Proof. The cell $\alpha(y; \tilde{z})$ may react only on y, but y is free, so the cell does not participate in any reduction of μ. □

For technical reasons, we introduce the following restricted notion of barbed bisimulation:

Definition 3.6.3 (x-bisimulation)**.** Let x be a port. An *x-bisimulation* is a binary relation $\mathcal{B}$ on nets of $\mathcal{S}$ such that, whenever $\mathcal{B}(\mu, \nu)$, $\mu \downarrow_x$ implies $\nu \Downarrow_x$ and $\nu \downarrow_x$ implies $\mu \Downarrow_x$, plus the usual reduction properties required by barbed bisimulations (last two points of Definition 3.1.10).

In other words, an x-bisimulation is a usual barbed bisimulation in which we content ourselves with simulating barbs on x only.

Lemma 3.6.4. *Let $\mathcal{B}$ be an x-bisimulation, and let $\mathcal{B}(\mu, \nu)$. Then, $\mu \Downarrow\!\!\Downarrow_x$ iff $\nu \Downarrow\!\!\Downarrow_x$.*

Proof. Immediate. □

Lemma 3.6.5. *Suppose that $\mathcal{S}$ is uniport and simply-wired, and let μ be a simply-wired net of $\mathcal{S}$ such that $\mu \Downarrow\!\!\Downarrow_x$. Then, for every simply-wired net ν such that $x \notin \mathrm{fp}(\nu)$, $(\mu \mid \nu) \Downarrow\!\!\Downarrow_x$.*

Proof. By definition, $\mathcal{O}_{\mathcal{S}} \neq \emptyset$, so let $(\alpha, 1, \beta, 1, k) \in \mathcal{O}_{\mathcal{S}}$. Let $\tilde{y}$ be a repetition-free sequence not containing x, of length equal to the number of auxiliary ports of α, and consider the relation

$$\mathcal{B} = \{(\mu \mid \nu, \alpha(x; \tilde{y})) \; ; \; \mu, \nu \text{ simply-wired}, \; \mu \Downarrow\!\!\Downarrow_x, \; x \notin \mathrm{fp}(\nu)\}.$$

We claim that $\mathcal{B}$ is an x-bisimulation. Let $(\mu \mid \nu, \alpha(x; \tilde{y})) \in \mathcal{B}$. First of all, $\mu \Downarrow\!\!\Downarrow_x$ implies $\mu \Downarrow_x$ which implies $(\mu \mid \nu) \Downarrow_x$, and by hypothesis $\alpha(x; \tilde{y}) \downarrow_x$, so the first two properties are met. Since $\alpha(x; \tilde{y})$ does not reduce, it is enough to show how $\alpha(x; \tilde{y})$ simulates a reduction $\mu \mid \nu \to \rho$. Such a reduction necessarily comes from an active pair ϕ. If ϕ is entirely contained in μ or ν, the definition of $\mathcal{B}$ allows us to conclude immediately. So we suppose that ϕ is an active pair created by the juxtaposition of μ and ν, *i.e.*, we may assume that

$$\mu \equiv \mu' \mid \beta(z; \tilde{t}),$$
$$\nu \equiv \gamma(z; \tilde{s}) \mid \nu',$$

with z free both in μ and ν, because both nets are simply-wired. Then, if ρ' is either $\beta_1 \blacktriangleleft\!\!\blacktriangleright \gamma_1\{\tilde{t}/\tilde{p}, \tilde{u}/\tilde{q}\}$ or $\gamma_1 \blacktriangleleft\!\!\blacktriangleright \beta_1\{\tilde{u}/\tilde{p}, \tilde{t}/\tilde{q}\}$ (for some irrelevant k), we have $\rho = \mu' \mid \rho' \mid \nu'$. But by Lemma 3.6.2, $\mu'\Downarrow_x$, so $(\rho, \alpha(x; \tilde{y})) \in \mathcal{B}$ by definition of $\mathcal{B}$.

Now, obviously $\alpha(x; \tilde{y})\Downarrow_x$ (as already observed, we have $\alpha(x; \tilde{y})\downarrow_x$ and the net does not reduce), so we may conclude by Lemma 3.6.4. □

Lemma 3.6.5 is false in presence of multiwires or multiports. For instance, consider an INS in which there are two symbols α, β, of degree 1 and 2, respectively, with the following interaction rule (which is observable, since it is the only one):

$$\alpha(x) \mid \beta(x; y) \to \alpha(y).$$

If we set $\mu = \alpha(x) \mid [x, y, z]$, we obviously have $\mu\Downarrow_y$ and $\mu\Downarrow_z$. However, for example, although still observable, z is no longer must-observable in $\mu \mid \beta(y; s)$, because $\mu \mid \beta(y; s) \to \alpha(s) \mid [z]$, in which there is no way to observe z. Similar examples may be built with multiports.

Theorem 3.6.6. *There exists an INS $\mathcal{S}$ which cannot be translated into any simply-wired, uniport INS $\mathcal{T}$ using only simply-wired nets.*

Proof. Take as $\mathcal{S}$ the system defined above, in which we allow nets containing multiwires, and suppose there exists a translation $[\![\cdot]\!]$ into a simply-wired, uniport INS $\mathcal{T}$ whose image consists of simply-wired nets only. Let $\mu = \alpha(x) \mid [x, y, z]$. Since $\mu \mathrel{\dot{\approx}} [\![\mu]\!]$, we must have $[\![\mu]\!]\Downarrow_z$. Consider now the net $\rho = \mu \mid \beta(y; s)$. By the homomorphism property, $[\![\rho]\!] = [\![\mu]\!] \mid [\![\beta(y; s)]\!]$. By port preservation, $x \notin \mathrm{fp}([\![\beta(y; s)]\!])$, so we may apply Lemma 3.6.5 (all nets in the image of the translation are simply wired) and infer that $[\![\rho]\!]\Downarrow_z$. But we saw above that we do not have $\rho\Downarrow_z$, contradicting the fact that $\rho \mathrel{\dot{\approx}} [\![\rho]\!]$. □

As already mentioned, although the system $\mathcal{S}$ used in the proof is uniport and uses multiwires, there is no difficulty in finding a simply-wired but multiport system $\mathcal{S}'$ for which Theorem 3.6.6 holds (with basically the same proof).

3.7 Comparing multiport and multiwire concurrency

We have seen until now two encoding results. We know that multiwires can absorb multiple rules and that cells with multiple principal ports are capable of absorbing both multirules (by splitting ports) and connectors (by duplicating ports). We have seen, on the other hand, that MP and MWR induce the same conflict graphs, so it might be possible to encode the first in the second. After all, those multiports seem to be exactly fit for a combination of multirules and connectors. Can they do anything more?

The last part of the previous section about encoding asymmetric reflexive rules into multiports surprisingly brings us an answer. We will not be able to respond entirely by the negative to this question, but we will see that even if such an encoding existed, it would be problematic. The section is dedicated to the following result:

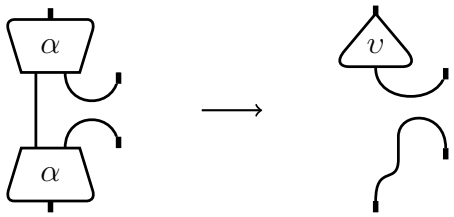
> There exists a self-rule-free MP system $\mathcal{S}$ which cannot be translated into any self-rule-free MWR system without introducing divergence.

The proof is based on the election problem, in a version very close to Palamidessi's result from [45]. Let us suppose we have an encoding from MP to MWR.

Let us consider the following MP system $\mathcal{S}$. Its alphabet has two symbols: α of coarity 2 and arity 1; υ of coarity and arity 1. $\mathcal{S}$ has two observable rules:

$$\alpha_1(a, s; y) \mid \alpha_2(a, t; z) \to \upsilon(y; s) \mid [t, z]$$

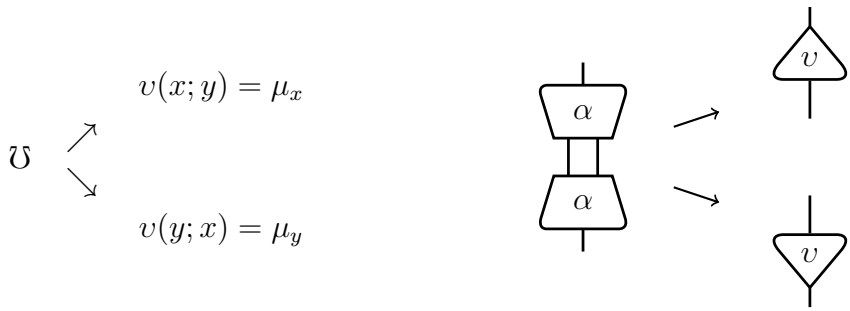
and any rule for $\upsilon_1 \bowtie \upsilon_1$. The idea is that when two α cells "meet", one on its first principal port, the other on its second, the one which interacts on the first "wins the election". The victory is represented by the fact that its auxiliary port becomes principal to a υ cell, which is observable by otherwise useless rule for $\upsilon_1 \bowtie \upsilon_1$. Graphically:



In that system, we consider the net

$$\mho = \alpha(a, b; x) \mid \alpha(b, a; y).$$

It has interface x, y, but no immediately observable ports, even though both ports of its interface are weakly observable. Indeed, whatever is connected to x or y, no interreductions are possible, but each of the two internal interactions makes one of x, y observable. The two possible internal reductions are



Nets μ_x and μ_y look alike but have opposite observational values: $\mu_x{\downarrow_x}$ but $\mu_x \not\Downarrow_y$ whereas $\mu_y{\downarrow_y}$ but $\mu_y \not\Downarrow_x$. Moreover, and it is important, μ_x and μ_y cannot reduce any

further. As a result, we have a system in which a net with a strong symmetry property can reduce in two ways into nets not having this property.

Definition 3.7.1 (Twofold net)**.** A net μ is *strictly twofold* if there exists a net ν whose free ports contain (but do not necessarily coincide with) $\tilde{s}, \tilde{t}, \tilde{u}$ such that

$$\mu = \nu\{\tilde{a}/\tilde{s}, \tilde{a}'/\tilde{t}, \tilde{x}/\tilde{u}\} \mid \nu\{\tilde{a}'/\tilde{s}, \tilde{a}/\tilde{t}, \tilde{x}'/\tilde{u}\}.$$

In that case, the free ports of μ are $\tilde{x}, \tilde{x}'$, and the pairs of ports x_i, x_i' are said to be *exchanged* by the symmetry. We say that μ is *twofold* if $\mu \equiv \mu_0$ with μ_0 strictly double.

Double nets formalize the fact that everything in them comes in pairs, cells, wires or ports, therefore barbs. In the absence of self-rules in a uniport framework, even active-pairs come in pairs. Moreover, being strict twofold is preserved by translation.

Lemma 3.7.2. *Let μ be a twofold and let $x, x' \in \mathrm{fp}(\mu)$ be exchanged by the symmetry. Then:*

- $\mu\downarrow_x$ *iff* $\mu\downarrow_{x'}$*;*
- $\mu\Downarrow_x$ *iff* $\mu\Downarrow_{x'}$ *.*

Proof. If $\tilde{y}, \tilde{y}'$ are the sequences of free ports of μ, with y_i, y_i' exchanged by the symmetry, then $\mu = \mu\{\tilde{y}'/\tilde{y}, \tilde{y}/\tilde{y}'\}$. The result then follows immediately. □

Lemma 3.7.3. *If μ is a strictly twofold and $[\![\cdot]\!]$ is a translation, $[\![\mu]\!]$ is strictly twofold too.*

Proof. An immediate consequence of the homomorphism and port invariance properties of translations. □

The key property twofold nets have in uniport systems is that they cannot break symmetry in one step without the use of a self-rule.

Lemma 3.7.4. *Let μ be a twofold net in a uniport INS that has no self-rules and let $\mu \to \nu$. Then, there exists a twofold net ν' s.t. $\nu \to \nu'$ (i.e. in 1 step).*

Proof. Let $\mu \equiv \rho\{\tilde{x}/\tilde{z}\} \mid \rho\{\tilde{x}'/\tilde{z}'\}$. Observe that the reduction $\mu \to \nu$ cannot concern an active pair created by the juxtaposition of the two copies of ρ. In fact, since ρ is uniport, such an active pair would consist of two cells carrying the same symbol. Then, the active pair φ reduced to obtain ν is entirely in one of the two components of μ and has a symmetric counterpart φ' in the other component. Because they are in separate components, φ and φ' do not overlap. Then, φ has a residue in ν; by reducing it (in the same way as the first one) we obtain a twofold net ν' . □

Theorem 3.7.5. *There exists a self-rule-free* MP *system $\mathcal{S}$ which cannot be translated into any self-rule-free* MWR *system without introducing divergence.*

Proof. We take as $\mathcal{S}$ the multiport system introduced at the beginning of the section and the net we denoted by $\mho$ (p. 140). Let $[\![\cdot]\!]$ be a translation of $\mathcal{S}$ into a uniport INS and let $\nu_0 = [\![\mho]\!]$. By Lemma 3.7.3, ν_0 is twofold and x, y are exchanged by its symmetry. By the bisimulation property, we know that there exists a barbed bisimulation $\mathcal{B}$ such that $\mathcal{B}(\mho, \nu_0)$. Since $\mho \to \mu_x$, we must have $\nu_0 \to^* \nu_x$ such that $\mathcal{B}(\mu_x, \nu_x)$. Since $\mu_x{\downarrow_x}$ but $\mu{\not\Downarrow_{xy}}$, by Lemma 3.7.2 we must have $\nu_x \neq \nu_0$, which means that at least one reduction step is possible from ν_0. Then, we may apply Lemma 3.7.4 and infer that $\nu_0 \to^* \nu_1$ in at least one reduction step, with ν_1 twofold. But this implies that $\mho \to^* \mu_1$ such that $\mathcal{B}(\mu_1, \nu_1)$. Now, μ_1 can only be one of μ_x, μ_y or $\mho$ itself, but the fact that ν_1 is twofold and Lemma 3.7.2 rule out the first two cases, hence $\mathcal{B}(\mho, \nu_1)$.

The reader is invited to check that, in the above reasoning, we deduced $\mathcal{B}(\mho, \nu_1)$ starting from $\mathcal{B}(\mho, \nu_0)$ using only the fact that ν_0 is twofold and that its two free ports are x, y (and must therefore be exchanged by its symmetry). These properties still hold for ν_1, so we may apply the reasoning again and again, obtaining a reduction sequence $[\![\mho]\!] = \nu_0 \to^* \nu_1 \to^* \nu_2 \to^* \cdots$, in which every reduction $\nu_i \to^* \nu_{i+1}$ is of length at least 1, so $[\![\mho]\!]$ diverges. □

Self-rules might not be the answer The absence of self-rules is primordial in Lemma 3.7.4. Let us place ourselves in a multiwire but uniport system with simple rules. This lemma is false in that setting: think for instance of the cell *qb* of Section 3.5.2. Such a rule *break the symmetry* in one step. It does not, nevertheless, break another symmetry in that same step, the one of observability.

We remind the definition of symmetric net.

Definition 3.7.6 (Symmetric net)**.** A net μ is *symmetric* if $\mu \equiv \mu\{\tilde{x}'/\tilde{x}, \tilde{x}/\tilde{x}'\}$ where $\tilde{x}, \tilde{x}'$ are the free ports of μ. x_i and x'_i are said to be *exchanged by the symmetry.*

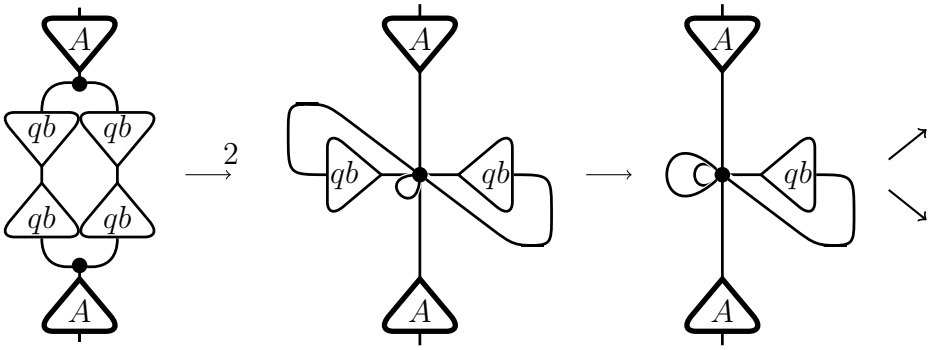
The equivalent of Lemma 3.7.2 is still valid for symmetric nets:

Lemma 3.7.7. *Let μ be symmetric and let $x, x' \in \mathrm{fp}(\mu)$ be exchanged by the symmetry. Then:*

- $\mu{\downarrow_x}$ *iff* $\mu{\downarrow_{x'}}$
- $\mu{\Downarrow_x}$ *iff* $\mu{\Downarrow_{x'}}$

Moreover, if μ is twofold and $\mu \to \nu$, then ν is symmetric. This is due to the fact that all simple rules are symmetric, which means exactly that if it is a self-rule, its RHS is a symmetric net. So the symmetry of observables can still not be "asymmetricized" in one step. Of course, it might be possible that any of the following steps necessarily breaks the observational symmetry. This is the way we simulate non-symmetric self-rules with multiwires. The difference here is that the rule we try to simulate is not a self-rule.

Let us consider the following wanabee-simulation of the net $\mho$.



It is not correct as an encoding as it can reach, from the middle state, a net in which both ports of the interface are observable indefinitely. At least as soon as the last state is capable of choosing. The problem is that a symmetric self-rule yields a net that is only capable of observing both or none, but not just one. Which is the case of our multiport system. If we now consider $\mho_x = \alpha(a, b_1; x) \mid \alpha(b_2, a; y)$. By just disconnecting the other active pair, this net is made *deterministic*: $\mho_x \Downarrow_x$ but $\mho_x \not\Downarrow_y$. If we consider the encoding above, the image of $\mho_x$ would not be barb bisimilar to it. Multiports have the capability of breaking a symmetry in two directions, or just in one. Which seems to not be the case in a uniport setting. Asymmetric self-rules do not seem to bring an answer either to that question.

About self-rules, asymmetric or not, they offer a great deal of questions and problems.

For instance, the abstract definition of barbs given by Rathke and als. [50] is problematic in the presence of self-rules. Based on orthogonality, where this later is given by the possibility of immediate reduction, in a testing flavor, self-rules lead to sets included in their orthogonal.

As about the expressivity there use can bring to a language, we have no clear idea. They are, to our knowledge, never used in the literature. As is, this problem cannot be addressed by translations, as their functional nature and homomorphism w.r.t. constructors necessarily translates a self-active pair into a net that has a self-active pair. If a study can be brought by encodings, it requires to encode wires into something different than wires. We have done this for the general encoding into multiports, but in languages which do not have the wire feature, as the usual name-based calculi languages, such an encoding is not possible. The search for a study of expressivity of self-rules will most probably not come from an encoding technique.

General election problem The impossibility for self-rule-free uniport systems to solve some model of the 2-election problem can be extended. In fact we are able to give a multiport net for any even natural p which is strongly twofold, but which computation ends in a non-symmetric net, necessarily.

We follow the general algorithm provided by Palamidessi in [45]. We build a net of the form $\mu_1 \mid \ldots \mid \mu_p$ in which all μ_i have the exact same form and fully interconnected two-by-two, with the following underlying procedure: each time a net μ_k *dominates* another net μ_l, the nets previously dominated by μ_l and μ_l itself are considered dominated by μ_k. Once a net is dominated, it is out of the game.

Let p be a natural number. We consider, for all $n < p$ a family of cells $\{\alpha_i^n\}_{2\leq i\leq 2n}$ with i principal ports and 1 auxiliary port. The subscript n of the label of a cell represents the amount of interactions this cell needs to win in order to win the entire election. By negative, it gives the amount of interactions this cell has already won: $p-n-1$. We also make use of a dummy cell ε with one unobservable principal port and the cell υ of the previous section.

The interesting interaction rules are the following:

$$\begin{aligned}\alpha_i^n(a_1,b_1,\ldots,a_k,b_k,\ldots,a_i,b_i;x) \mid \alpha_j^m(a'_1,b'_1,\ldots,a'_l,b'_l,\ldots,a'_j,b'_j;y) \mid [a_k,b'_l] \\ \rightarrow\ \alpha_{i-2}^{n-(p-m)}(a_1,b_1,\ldots,a_{k-1},b_{k-1},a_{k+1},b_{k+1},\ldots,a_i,b_i;x)\ \mid\ \varepsilon(b_k) \\ \mid\ \prod_{\substack{z=1\\z\neq l}}^{j}(\varepsilon(a'_z)\mid\varepsilon(b'_z))\ \mid\ [a'_l,y]\end{aligned}$$

iff $n-(p-m)\neq 0$, else

$$\begin{aligned}\alpha_i^n(a_1,b_1,\ldots,a_k,b_k,\ldots,a_i,b_i;x) \mid \alpha_j^m(a'_1,b'_1,\ldots,a'_l,b'_l,\ldots,a'_j,b'_j;y) \mid [a_k,b'_l] \rightarrow \\ \upsilon(x;b_k)\ \mid\ \prod_{\substack{z=1\\z\neq k}}^{i}(\varepsilon(a_z)\mid\varepsilon(b_z))\ \mid\ \prod_{\substack{z=1\\z\neq l}}^{j}(\varepsilon(a'_z)\mid\varepsilon(b'_z))\ \mid\ [a'_l,y]\end{aligned}$$

which generalizes the rule we used for the separation result. Ports of α-cells can be erased by ε-cells, but it is not needed for the correctness of the algorithm. As usual, two ε-cells erase each other when they interact.

We claim that the net Λ_p we define below is such that any reduction is finite and such that, for any net μ such that $\Lambda_p \rightarrow^* \mu \not\rightarrow$, there exists $1\leq k\leq p$ such that $\mu\Downarrow_{x_k}$ but $\mu\Downarrow_{x_{k'}}$ for all $k'\neq k$.

$$\begin{aligned}\Lambda_p = \alpha_{2p-2}^{p-1}(a_1^0,b_1^0,\ldots,a_{p-1}^0,b_{p-1}^0)\mid\ldots\mid\alpha_{2p-2}^{p-1}(a_1^{p-1},b_1^{p-1},\ldots,a_{p-1}^{p-1},b_{p-1}^{p-1})\mid \\ \prod_{k=0}^{p-1}\prod_{i=1}^{p-1}[a_i^k,b_{p-i}^{(k+i)\mod p}]\end{aligned}$$

Moreover, Λ_p is strictly twofold. A possible reduction path of Λ_4 is shown in Figure 3.9

Theorem 3.7.8. *For any even natural p, there is a multiport system $\mathcal{S}_p$ and a net Λ_p*

in that system which cannot be translated into any self-rule-free MWR *system without introducing divergence.*

In fact, the twofoldness of a net is a particular case of cycle. The same result could be obtained by considering a cyclic automorphism of any power $n \leq 2$, leading to the impossibility to encode any Λ_n of a system $\mathcal{S}_n$, for any natural $n \leq 2$. For a better understanding of how such a property is defined, we address the reader to the paper that inspired this result and the election problem solving algorithm [45].

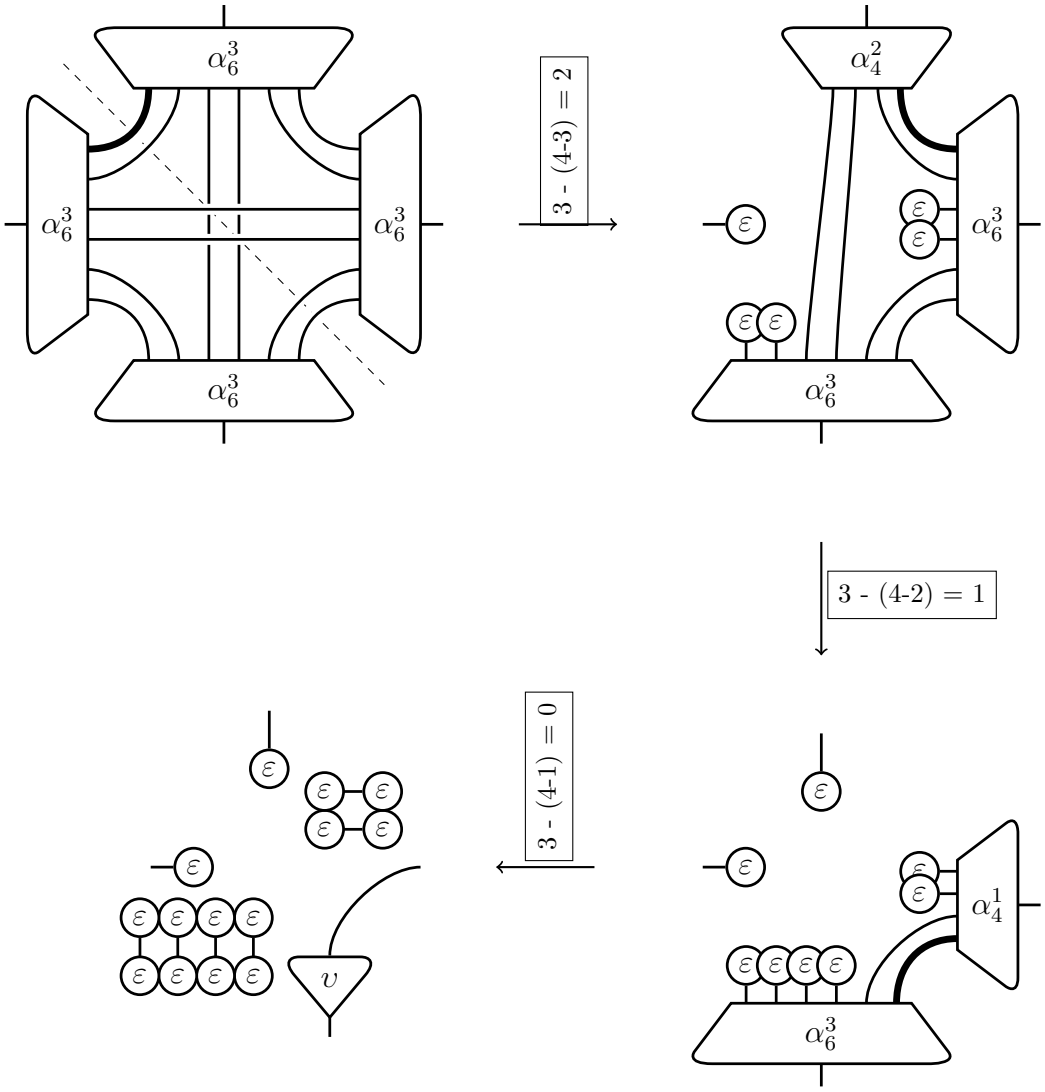


Figure 3.9: A possible reduction path for Λ_4 in $\mathcal{S}_4$.
The active pair for the reduction is shown in bold. The operation determining the counter of how many cells remain to be dominated is given with every interaction : counter of winner − (p − counter of loser).

Chapter 4

Multiport Combinators

In this last chapter, we introduce a universal system for strictly multiport interaction nets, *i.e.* simply wired with simple rules. Since we have proven in the previous chapter that any interaction net system can be encoded into one of these, the *multiport combinators* we present can be considered as a universal system for general interaction nets.

The procedure relies on the ability to decompose a net, namely the right-hand-side of a rule, into two nets which have no active pairs and no deadlocks, making it possible to erase and most of all duplicate them. A splitting of this sort is discussed in Section 4.1. Section 4.2 presents the language of multiport combinators and some constructions that allow to encode system without recursion. Such an encoding is given in Section 4.3. The general case of encoding system with recursion is detailed in Section 4.4, after describing the duplication procedure (4.4.1). A last short section (4.5) question the quality of the system.

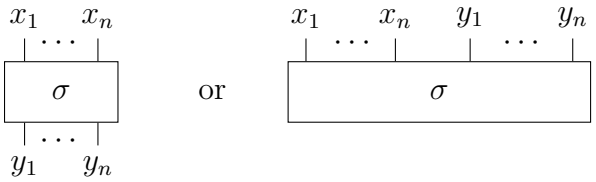
The concepts for this chapter are introduced in the paragraph about universality (p. 14).

4.1 Special decomposition of nets

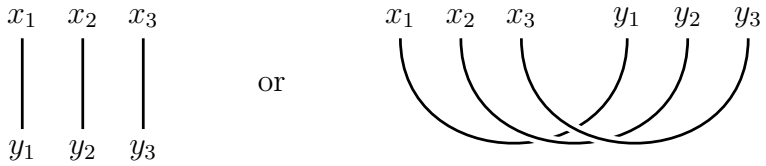
We start by given a useful decomposition of nets. It is not at all natural as in Lafont nets, as the structure of multiport nets is really messy, geometrically speaking. We will use this decomposition to split RHS s of rules in two special kinds of nets, that basically have no active pairs. We also use this decomposition to show that some special nets, called *reduced*, can be duplicated when necessary.

Let us start by some useful vocabulary.

Definition 4.1.1 (Lafont [33])**.** A *wiring* is a net ω without cells and cyclic wires. So it is just a pairing of its free ports. In particular, a wiring has an even number of free ports. A permutation σ of $\{1, \ldots, n\}$ defines a wiring with $2n$ free ports, which is represented as follows:



In both cases, x_i is connected to $y_{\sigma(i)}$. For instance, the wiring corresponding to the identity on $\{1, 2, 3\}$, $[x_1, y_1] \mid [x_2, y_2] \mid [x_3, y_3]$ is pictured as follows:



As we have seen, we have to be careful about reflexive rules and symmetric nets. Luckily, in strictly multiport systems, symmetric nets have a precious feature.

Lemma 4.1.2. *Let $\mathcal{S}$ be an IN and μ a symmetric net with only simple wires. Then μ is twofold.*

Proof. The proof is by induction on the size of the net, where the size means the amount of wires and cells.

If $\mu \equiv [a, b]$, then it is twofold.

If $\mu \equiv \mu\{x'/x, x/x'\}$, we can check that, for any i, either x_i, x_i' belong to a same wire (or series of wires) or are connected to the same principal or auxiliary port of cells c, c' of same label, thus of two different cells. In the first case, if we remove the wire (which is a twofolded net), we obtain a smaller net. In the second cases, we remove the two cells x_i and x_i' are connected to, obtaining also a smaller net. □

In the proof, when x_i, x_i' are connected to cells, we consider the two cells to also *be exchanged by the symmetry* of the net.

Unlike in Lafont nets, characterizing deadlocks is a complicated matter in a multiport setting. We give a general definition:

Definition 4.1.3 (Deadlock)**.** A *deadlock* is a net with no (reducible) active pairs and such that all principal ports are bound.

In this chapter, we will call *cut* any kind of configuration that reminds an active pair, even in an unusual way:

Definition 4.1.4 (Cut)**.** We call *cut* any net composed of one or two cells that contains a wire which connects two principal ports. When needed, we call *irreducible cut* a cut which is not an active pair, so active pairs are sometimes referred to as *reducible cuts*.

Cuts on a single cell are dangerous as they can create deadlocks. Imagine simply a cell of coarity 2 and a wire connecting its two principal ports. But we also want to avoid situations like an $\alpha(a, b, c; d) \mid [a, b]$. In fact, we would not be able to guarantee that a net containing too many of those can be duplicated.

We remind, to be sure to be precise, what is a subnet.

Definition 4.1.5 (Subnet)**.** A subnet N of a net M is a subset S of the set of cells of M and contains at least all ports of cells in S. There is no condition on wires or other ports.

The first important types of nets we need are the following:

Definition 4.1.6 (Reduced net)**.** A net N is *reduced* if none of its subnets are neither cuts nor deadlocks.

An irreducible net is not necessarily reduced, as it can contain irreducible cuts or deadlocks, but a reduced net is always irreducible of course.

Let us make a remark here. Any closed net, *i.e.* with no interface, is barbed congruent to the empty net. This is due to the fact that bound ports can never be made free, and rules cannot create new free ports. Furthermore, since observability is built on interreduction, no free ports means no possible observable, ever. This means that even if the closed net is not irreducible, worse, even if it contains observable active pairs, it is barbed congruent to the empty net.

Unlike in Lafont nets, we do not need to consider RHS of rules to be deadlock free, or even cut-free. In fact, it is sometimes useful to have reducible RHS, as for instance for [20]. Because of the absence of confluence, there is in fact no argument to consider that rules produce only reducible nets.

We now define some special reduced nets. They are a generalization of trees to a world with multicells.

Definition 4.1.7 (Multitree)**.** A wire $[a, b]$ is a *multitree*. It can be considered in two ways. Either its *root interface* is a, in which case b is its *leaf interface*, or both a and b are in its leaf interface, in which case its root interface is empty.

Let $T_1(\tilde{b}_1, \tilde{z}_1), \ldots, T_n(\tilde{b}_n, \tilde{z}_n)$ be n trees with resp. $\tilde{b}_i$ as root interface and $\tilde{z}_i$ as leaf interface. Let $\tilde{a}$ be a sequence of fresh names and $\tilde{x}$ such that

$$\tilde{x} \cap \tilde{z}_i = \emptyset \quad \forall 1 \leq i \leq n \qquad \text{but} \qquad \tilde{x} \cap \tilde{b}_i \neq \emptyset \quad \forall 1 \leq i \leq n.$$

Then $\alpha(\tilde{a}; \tilde{x}) \mid T_1(\tilde{b}_1, \tilde{z}_1) \mid \ldots \mid T_n(\tilde{b}_n, \tilde{z}_n)$ is a *multitree*, of root interface any subsequence a' of a (even empty) and leaf interface all other free ports (*i.e.* $a \setminus a'$, $b_i \setminus (b_i \cap \tilde{x})$ and $\tilde{z}_i$). The cell labeled α is its *root*.

Let $T(\tilde{a}, \tilde{z})$ be a multitree of root interface $\tilde{a}$ and leaf interface $\tilde{z}$, then $T(\tilde{a}, \tilde{z}) \mid [z_i, z_j]$ is a *multitree* of same root interface and root but of leaf interface $\tilde{z} \setminus \{z_i, z_j\}$.

Any net structurally congruent to a multitree is a multitree.

What we achieve with this complicated definition is to get a net which looks like a tree if you consider cells as nodes and vertices to be the existence of a wire connecting an auxiliary port of a cell to a principal port of another. A representation of a generic multitrees is represented in Figure 4.1. It can be misleading, since some ports on top can be principal ports of some cells inside. We will remind the reader this fact when necessary.

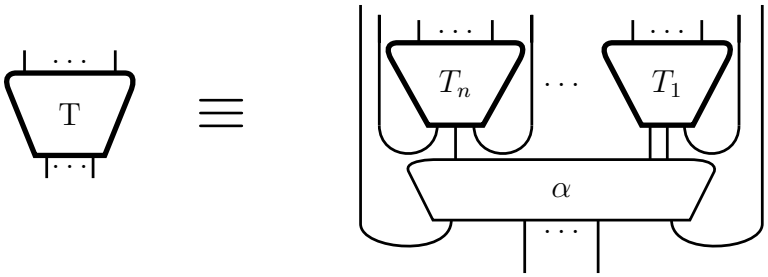
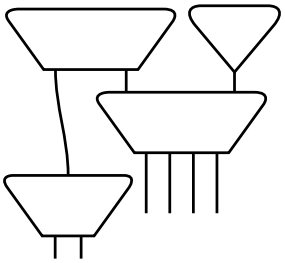


Figure 4.1: How a generic multitree looks like.

We need for leaf interfaces to grab every principal port of cells that are not the root in order to avoid having "several roots" like in the dummy example below. Such a situation is not problematic for the decomposition lemma but forbids duplication.



We call just *tree* a multitree in which all cells are simple and the root interface is non-empty, *i.e.* of size 1 exactly. Notice that in a tree, all ports of the leaf interface are auxiliary ports.

Proposition 4.1.8. *A multitree is a reduced net.*

Lemma 4.1.9 (Multitree decomposition)**.** *Let N be a net. It is possible to decompose N into n multitrees and a wiring on the union of their leaf interfaces.*

Proof. The procedure is simple. Pick randomly a cell. It is going to be the root of the first multitree. Those of its principal ports that are free form the trees root interface. Its other principal cells are part of the leaf interface. Those of its auxiliary ports that are connected to an auxiliary port of another cell or are free are part of the trees leaf interface. Those of its auxiliary ports connected to an auxiliary port of the tree (for now just the cell) are part of the tree, with the wires connecting them. If an auxiliary port is connected to a principal port of the cell itself, it is considered as part of the leaf interface. The others are connected to principal ports of some other cell. Those cells are "added" to the tree and processed in the same way, except that all their principal ports not connected to auxiliary ports of the root are part of the leaf interface.

The procedure ends when all auxiliary ports of all cells are part of the leaf interface. The first tree is finished.

Pick a cell that is not part of this tree as a root for a second tree. Etc.

At the end, connect two by two all ports of the leaf interface as they connected ports in N. For any port from the leaf interface that is free, add a wire (which is a

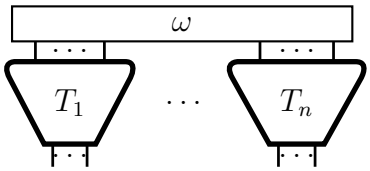
tree) connected to that port with its leaf. Its other port is considered its root interface, and is part of the interface of N. □

Since cells alone make multitrees, we can even consider a maximal decomposition, cell by cell, taking then care of the whole wiring of the net. The decomposition is not at all unique and depends on what cells are chosen as roots. Unlike in Lafont nets, there is no striking canonical choice for roots of the trees. The trick nevertheless relies on the fact that a wire can be split in many wires by structural congruence. We will use that trick often in the following proofs.

Corollary 4.1.10. *Let $N(\tilde{a}, \tilde{b})$ be a symmetric net which symmetry exchanges $\tilde{a}$ and $\tilde{b}$. There exists a net M s.t. $N(\tilde{a}, \tilde{b}) \equiv M(\tilde{a}, \tilde{z}) \mid M(\tilde{b}, \tilde{z})$.*

Proof. Proceed in the same way but always building in parallel two trees with roots cells that are image to one another by the symmetry. □

Corollary 4.1.11. *Any reduced net N can be decomposed as follows:*

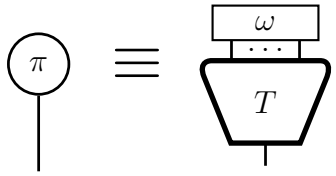


where $T_1, \ldots, T_n$ are multitrees and ω is a wiring. In other words, there is a multitree decomposition of N in which all multitrees have nonempty root interface.

Proof. In fact, if N is reduced and one of the multitrees in its decomposition, let us call it T, has an empty root interface, then all the principal ports of its root cell are connected to an auxiliary port of other cells. If all these cells already belong to T, then T is a deadlock, which is impossible. So one of these cells belong to another multitree T'. Then, it is sufficient to build a bigger multitree with same root as T' but containing also T, which keeps the decomposition correct.

Remember of course that a simple wire is a multitree, so the interface of N can contain auxiliary ports. □

We can see by this decomposition that a reduced net with no free ports is necessarily empty. A reduced net with one free port (which is therefore principal) is called a *package*. It has the following decomposition:



where T is a multitree with a unique port in its root interface and ω a wiring.

Most nets of course are not reduced. We now show a way how to split any net into two part connected by a wiring, each of the parts being reduced.

Lemma 4.1.12 (Splitting of a net)**.** *Let N be a net and I_1, I_2 a two-partition of its interface. Then there exist two reduced nets M_1, M_2 such that $\overline{I}_1 \subset M_1$, $I_2 \subset M_2$ and a wiring ω on $(\mathrm{fp}(M_1) \setminus I_1), (\mathrm{fp}(M_2) \setminus I_2)$ such that*

$$N \equiv M_1(I_1, \tilde{y}) \mid M_2(I_2, \tilde{z}) \mid \omega(\tilde{y}, \tilde{z}).$$

We call M_1 and M_2 halves of N along I_1 and I_2. Moreover, if N is symmetric, then it is possible to find a decomposition in which $M_1 \equiv M_2$.

Proof. Partition the net by Lemma 4.1.9 into a certain number of multitrees and assign each of the trees randomly to M_1 and M_2. To any port of a root interface of one of the trees that is free add a wire between it and the free port. For each wire $[x, y]$ of N that is not already in M_1 or M_2 (they should be the ones in the wiring of the decomposition or connecting a port of a root interface of a tree to a free port).

- if $x, y \in I_1$ (resp. I_2), then assign $[x, y]$ to M_1 (resp. to M_2).
- if $x \in I_1, y \in I_2$ (or vice-versa), then $[x, y] \equiv [x, z_1] \mid [z_1, z_2] \mid [z_2, y]$ and assign $[x, z_1]$ to M_1 and $[z_2, y]$ to M_2. $[z_1, z_2]$ will be part of ω.
- if $x \in I_1$ and $y \in M_1$ (resp. I_2 and M_2), assign $[x, y]$ to M_1 (resp. M_2).
- if $x \in I_1$ and $y \in M_2$ (resp. I_2 and M_1), then $[x, y] \equiv [x, z_1] \mid [z_1, z_2] \mid [z_2, y]$ and assign $[x, z_1]$ to M_1 and $[z_2, y]$ to M_2. $[z_1, z_2]$ will be part of ω.
- if $x \in M_1$ and $y \in M_2$, then $[x, y] \equiv [x, z_1] \mid [z_1, z_2] \mid [z_2, y]$ and assign $[x, z_1]$ to M_1 and $[z_2, y]$ to M_2. $[z_1, z_2]$ will be part of ω.
- if $x, y \in M_1$ (resp. M_2), then $[x, y] \equiv [x, z_1] \mid [z_1, z_2] \mid [z_2, z_3] \mid [z_3, z_4] \mid [z_4, y]$ and assign
  - $[x, z_1]$ and $[z_4, y]$ to M_1 (resp. M_2);
  - $[z_2, z_3]$ to M_2 (resp. M_1);
  - $[z_1, z_2]$ and $[z_3, z_4]$ are part of ω.

If N is symmetric, thus twofold, we do not partition the net randomly but along its "symmetry". The procedure gives the desired result. □

Graphically, the different configuration of wire-splitting are shown in a graphical representation in Figure 4.2. Thanks to the last point, any possible cut is avoided in each of M_1 and M_2 since (at least) every wire connecting any principal ports is transformed into a wire that passes through the other *half*. Actually, M_1 and M_2 each composed of disjoint multitrees. This decomposition is again far from being unique.

The reader must be careful not to mix-up the different decompositions. In the splitting of a net each half of N is reduced, but its interface is both *in front* and *in the back*. The wiring given in the splitting is a wiring *between the two parts of the splitting*. In the tree decomposition on the other hand, all the interface was *brought to the front* with the help of degenerated multitrees (wires) and a wiring *that belongs to the net*.

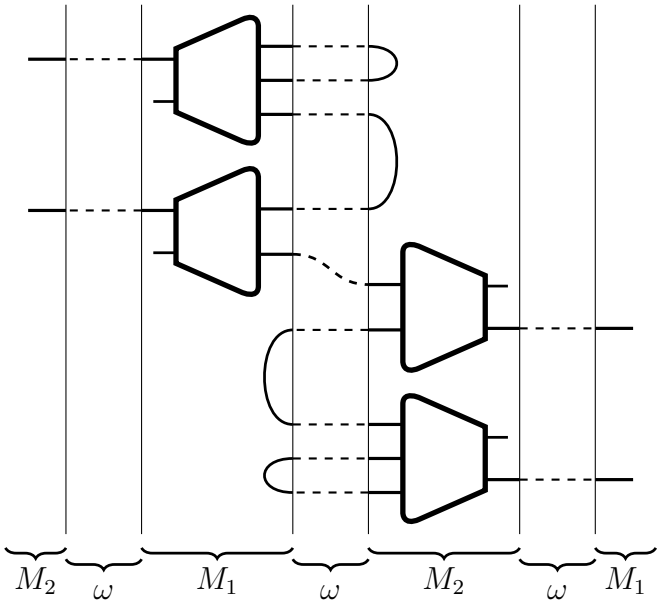


Figure 4.2: All situation for splitting wires outside of multitrees.
Remember that ports on the back of multitrees are not necessarily auxiliary.

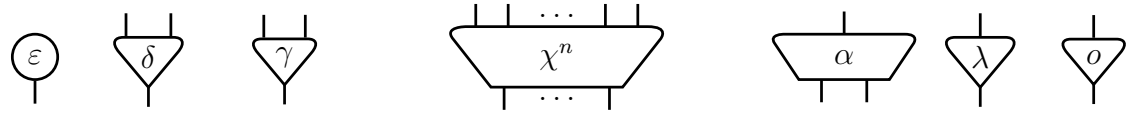
4.2 Combinators for multiport interaction nets

In this section we use a lot of results of interaction combinators of Lafont for simple nets as they are (of course) still useful in the multiwire case. In order not to overload the reading, we will not notify every result that it already in Lafont's work, but it is not our wish to take credit for any thing that was already in his work.

4.2.1 The system

The system of multiport combinators, $\mathbb{MC}$ for short, contains a denumerable set of cells, but only a finite number is needed to encode any multiport system $\mathcal{S}$. Six cells are distinguished, which are *fixed*, and an infinite family is used to express all possible coarities of the original system. If the largest coarity in $\mathcal{S}$ is n, then only n cells of the infinite family are needed in the encoding.

The cells of $\mathbb{MC}$ are the following:



$\varepsilon(a)$ $\delta(a;x,y)$ $\gamma(a;x,y)$ $\alpha(a,b;x)$ $\lambda(a;x)$ $o(a;x)$

$$\chi^n(a_1,\ldots,a_n;x^1,y^1,\ldots,x^n,y^n)$$

The first three are Lafont's interaction combinators. Their respective purpose is

to *erase* cells, *duplicate* cells and *exchange* ports[1]. The family $\{\chi^n\}_{n\in\mathbb{N}}$ are cells that control the number of principal ports in the interface of the net that simulates a given cell. α plays two roles. Against γ-cells, it plays the role of a resource allocator. Against χ-cells it blocks interaction to simulate inactive pairs. λ is then needed to annihilate the blocker α if the transformation of one of the cells of the inactive pair has transformed the cut into an active pair.

We use the label θ for generic (multi)cell label.

Duplicator δ The cell labeled δ is mainly used to duplicate all other cells. Its general rule is therefore:

$$\delta(a;x,y) \mid \theta(a_1,\ldots,a_n) \mid [a,a_k] \to \theta(b_1,\ldots,b_n) \mid \theta(b_1,\ldots,b_n) \mid \prod_{\substack{i=1\\ i\neq k}}^{n} \delta(a_i;b_i,c_i)$$

for any label θ different from δ, and any k such that k-th port of θ is principal.

In order to duplicate entire nets, we need for it to annihilate itself, otherwise a duplication would lead to infinitely growing nets.

$$\delta(a;x,y) \mid \delta(a;x',y') \to [x,x'] \mid [y,y'].$$

None of the rules for δ are observable. Graphically, they are shown in Figure 4.3(a).

Erasor ε The cell labeled ε is used as garbage collecting in almost all cases. Its usual way of acting is to erase the other cell c and send ε cells on all other ports of c, principal or auxiliary.

$$\varepsilon(a) \mid \theta(a_1,\ldots,a_n) \mid [a,a_k] \to \prod_{\substack{i=1\\ i\neq k}}^{n} \varepsilon(a_i)$$

for any label θ different from α, and any k such that k-th port of θ is principal. In fact, α is an exception. In this case, the cell labeled α is still erased, but the two remaining ports are linked together:

$$\varepsilon(a_1) \mid \alpha(a_1,a_2;x) \to [a_2,x]$$
$$\varepsilon(a_2) \mid \alpha(a_1,a_2;x) \to [a_1,x]$$

None of the rules for ε are observable. Graphically, they are shown in Figure 4.3(b). Note that $\varepsilon \bowtie \delta$ is defined uniquely, as the result is the same if you look at it as a duplication of ε or erasing of δ.

[1]Pretty much anything that implies only those cells is already in [33]. Anything with the others cells is original to this work

Exchanger γ The real important rule for γ is its reflexive rule, in which it exchanges auxiliary port numbers:

$$\gamma(a; x_1, x_2) \mid \gamma(a; y_1, y_2) \to [x_1, y_2] \mid [x_2, y_1].$$

Notice that the symmetry condition is verified. This rule is never observable. Another rule is its interaction with α, given below.

Strange cell α As we have said before, α is the combination of two very different behaviors. When it meets a γ-cell, this last one *wins* access to the auxiliary port, while an eraser is sent to its other principal port.

$$\alpha(a, b; x) \mid \gamma(a; y, z) \to \gamma(x; y, z) \mid \varepsilon(b)$$
$$\alpha(a, b; x) \mid \gamma(b; y, z) \to \gamma(x; y, z) \mid \varepsilon(a)$$

On the other hand, when it meets a λ cell, it means it is a *blocker* which block access from one of its principal ports to the other, in which case λ *unblocks* it:

$$\alpha(a, b; x) \mid \lambda(a; y) \to \lambda(b; y) \mid \varepsilon(x)$$
$$\alpha(a, b; x) \mid \lambda(b; y) \to \lambda(a; y) \mid \varepsilon(x)$$

None of the rules for α are observable. Graphically, they are shown in Figure 4.3(d).

Unblocker λ and observer o In simple interaction nets, an irreducible cut automatically created deadlock. Simulating one of them could be done by reducing its encoding to any deadlock. In the multiport setting, it is not the case anymore, so irreducible cuts have to be simulated by irreducible cuts. That is why we need a blocking mechanism.

Most all the rules for λ have been already given: λ can be duplicated, erased, and most of all, it can erase a blocker α-cell. It does not need to interact in any particular way with γ-cell, but it needs to disappear in case it meets a χ-cell, as it means the *maybe active-pair* is no more to be:

$$\lambda(a; x) \mid \chi^n(a_1, \ldots, a_n; \tilde{z}) \mid [a, a_k] \to \chi^n(a_1, \ldots, a_{k-1}, x, a_{k+1}, a_n; \tilde{z}).$$

There are two options for the reflexive rule for λ, and both lead to a correct encoding: they can commute or get erased. We tossed a coin and chose the second version.

$$\lambda(a; x) \mid \lambda(a; y) \to [x, y].$$

Strangely also, a cell like λ will be needed to create observability. As it is only when a long part of the encoding process has started that one can know if this simulation

is one of an observable interaction or if it is one of a silent interaction, or worse, an inactive pair. It is at this point that we need to be able to observe interaction or not. So λ has a brother cell o, which has the same reflexive rule but which is observable.

Graphically, the rules for λ and so o are shown in Figure 4.3(e).

The χ^n family Here again we have almost given all the rules for these cells: they can be erased, duplicated and they can get rid of extra unblocker cells. There is no need for any interaction with γ-cells as this will never need to happen. It is on the other hand very important that there is no rule for active pairs of χ^n and α. In fact, this will allow us to block some cuts to behave like inactive pairs.

The most important rule anyhow for this family of rules are the one among themselves. The main idea of the encoding is that a cell of coarity n is encoded into some net containing a χ^n-cell in its *front*, representing all its principal ports. When two such cell interact, they trigger the simulation of the corresponding interaction. The purpose of universality for combinator makes it necessary for any such created configurations to trigger interaction. We give the intuition about this rule later. We now just give the formal rule, even though the reasons of such a rule are not yet clear:

$$\chi^n(a_1,\dots,a_n;s_1,t_1,\dots,s_n,t_n) \mid \chi^n(b_1,\dots,b_m;x_1,y_1,\dots,x_m,y_m) \mid [a_i,b_j]$$
$$\to \quad [s_i,x_j] \mid [t_i,y_j] \mid \prod_{\substack{k=1\\k\neq i}}^{n}\Big([a_k,t_k] \mid \varepsilon(s_k)\Big) \mid \prod_{\substack{k=1\\k\neq j}}^{m}\Big([b_k,y_k] \mid \varepsilon(x_k)\Big)$$

Graphically, the rules for α are shown in Figure 4.3(f). We will discuss the observability of these rules further on.

As a result – if we count the whole family of χ-cells as one – multiport combinators have 7 cells, out of which 5 are simple. Since the rules for multicells χ^n and α do not depend on which port is involved, we consider rules to be well defined when giving a pair of cells alone, without specifying the ports involved. This means 28 possible pairs, out of which "only" 22 are active pairs. Rules can be added without danger for some of the inactive pairs, except for α vs. χ^n, as this pair will be used to simulate inactive pairs from the original system. Cuts between γ and λ as well as between γ and χ^n should never occur during the encoding. Cuts α/α happen and can be given several different rules compatible with the blocking process, but it is not necessary.

In order to explain the encoding gradually, we give some particular feature (multiport) interaction nets can have. We then get rid of these features one by one to get more and more general.

As we know, some cut-like configurations can make inactive pairs. We qualify of *full* an interaction net system in which all pairs on principal ports have an interaction rule. If, furthermore, all its rules are observable, it is qualified *transparent*.

Another quality systems can have has more to do with computational capabilities: possibility of recursion – a rule for $\theta \bowtie \theta'$ can use or not cells θ or θ' in its RHS. In fact, we need to be even more restrictive. An order should be given on cells, and a RHS can contain or not cells greater for that order than the ones of the corresponding active pair.

We will first consider full and transparent recursion-free systems. Encoding such a system $\mathcal{S}$ does not require the blocking procedure – so no λ cell –, neither the use of the observable cell.

We then consider *opaque* systems, in which some rules are not observable. The trick is then really simple: the encoding of an observable rule creates an observable active-pair, and other encodings do not.

We go on considering system in which not all pairs are active. Then, the encoding of an inactive pair reduces to itself, since other ports of these cells may want to interact. In order to avoid divergence, this newly created pair which is not active is connected by a net that represents a blocked wire. Any other reduction involving one of these two cells unblocks the wire. This is done with the help of α and λ.

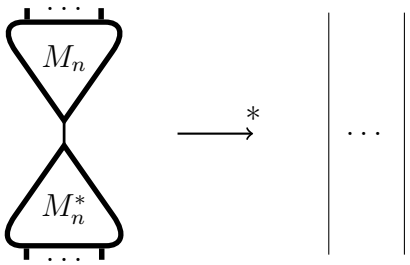
Finally, we show the most interesting part: how to encode recursion, when a active-pair creates cells greater than itself, creating the possibility of infinite reduction.

From now on, and until further notice, each rule $\chi_i^n \bowtie \chi_j^m$ is considered observable, for any $m, n, i, j \in \mathbb{N}$. All other rules are not.

4.2.2 Multiplexors and transpositors

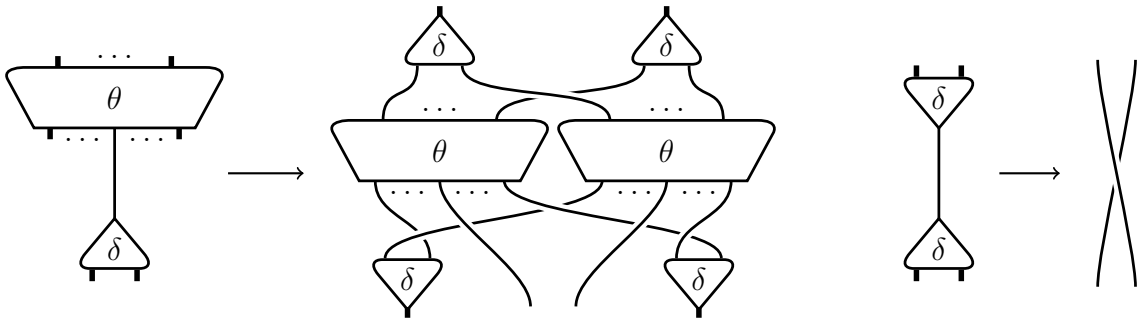
Multiplexors and transpositors are taken as is from Lafont's work.

For any $n \in \mathbb{N}$, one constructs two trees M_n and M_n^* (*multiplexors*) with n ports in the leaf interface (thus, all auxiliary), with the following property:

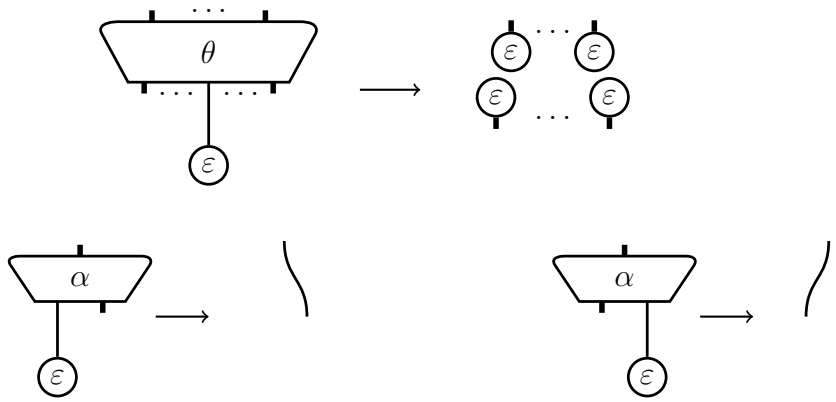


An implementation of these multiplexors is given in Figure 4.4. The needed rules are $\varepsilon \bowtie \varepsilon$ and $\gamma \bowtie \gamma$. There are alternative implementations using δ or χ^1 instead of γ, but it will be essential that the implementation of the multiplexors does not use δ, and the version with χ^1-cells is less *drawable*.

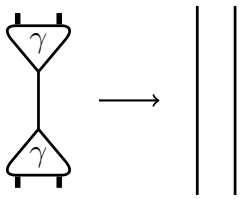
We shall also need a kind of autodual multiplexor, that is, a tree T_n with n ports in its leaf interface with the following property:



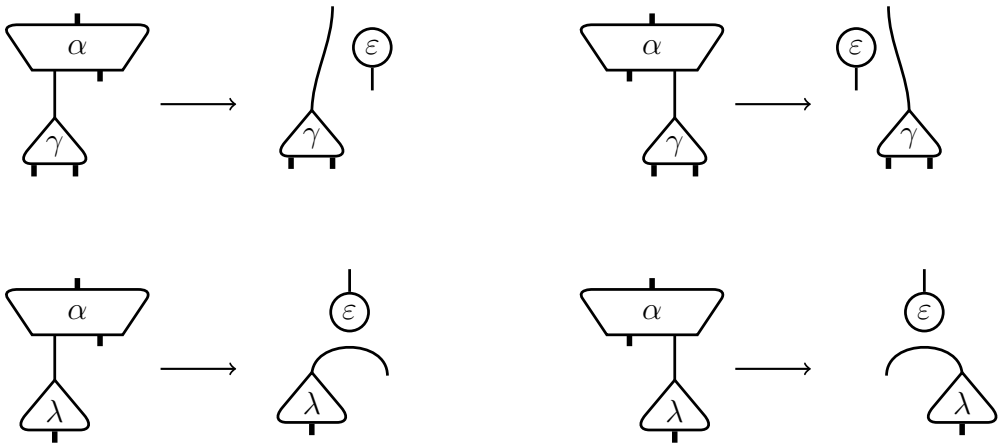
(a) Rules for δ



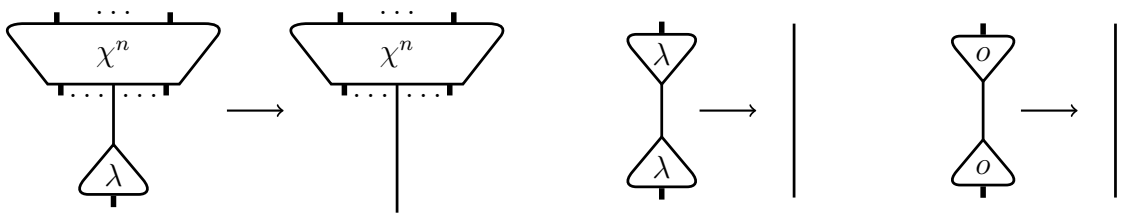
(b) Rules for ε



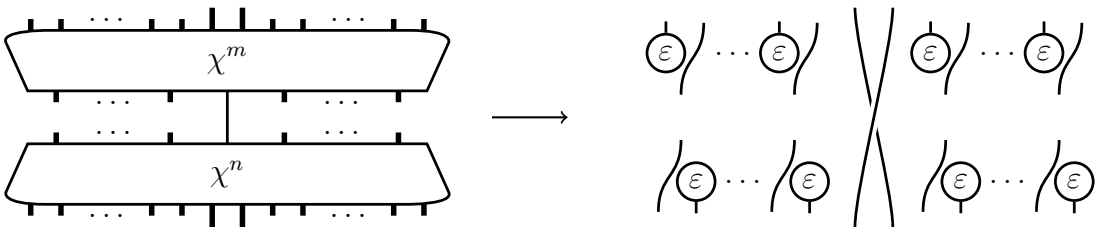
(c) Reflexive rule for γ



(d) (Remaining) rules for α

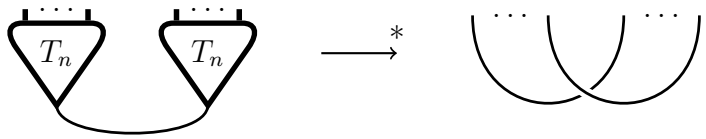


(e) (Remaining) rules for λ and observable rule $o \bowtie o$



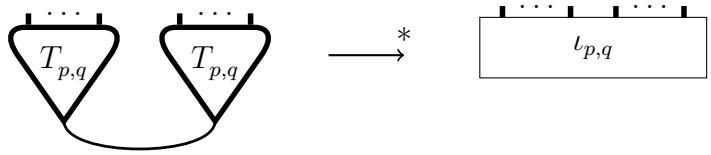
(f) The (sometimes observable) rule for the χ family

Figure 4.3: Multiport interaction combinators

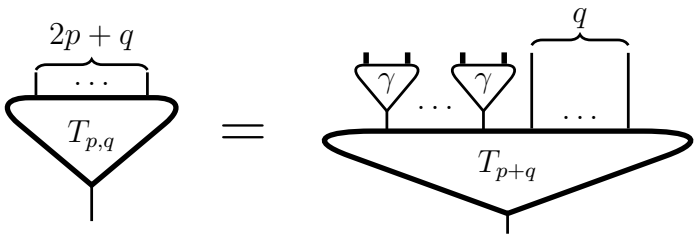


An implementation of these autodual multiplexors is given in Figure 4.5. The needed rules are $\varepsilon \bowtie \varepsilon$ and $\delta \bowtie \delta$. χ^1 can replace δ, but γ is not at all suitable for that purpose.

More generally, if $p, q \in \mathbb{N}$, one constructs a tree $T_{p,q}$ (*transpositor*) with $2p+q$ leaf ports, with the following property:

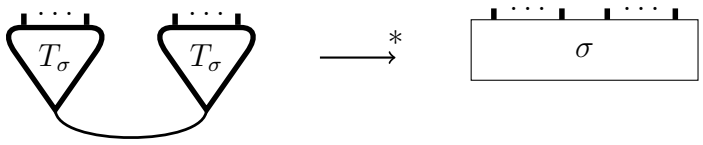


where $\iota_{p,q}$ is the involutive permutation of $\{1, \ldots, 2p+q\}$ which exchanges 1 with 2, 3 with 4 and so on until $2p$. Here is a possible implementation of these transpositors:

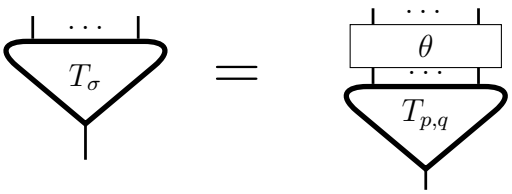


The needed rule is $\gamma \bowtie \gamma$. Note that δ and χ^2 are not suitable at all for that purpose.

Finally, if σ is an involutive permutation of $\{1, \ldots, n\}$, one constructs a tree T_σ with n ports in the leaf interface with the following property:



Any such permutation is indeed a product of disjoint transpositions, that is $\sigma = \theta^{-1} \circ \iota_{p,q} \circ \theta$ where θ is a permutation of $\{1, \ldots, n\}$ and $2p + q = n$. So a possible implementation of T_σ is:



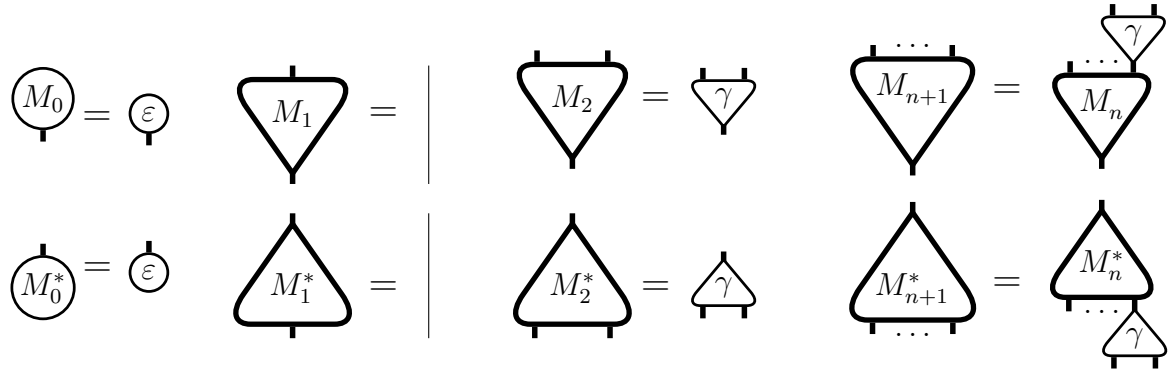


Figure 4.4: Implementation of the multiplexors

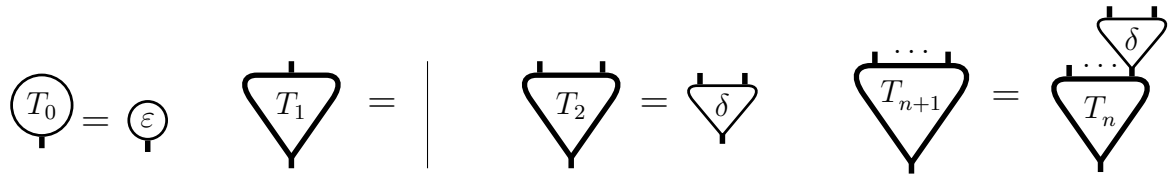


Figure 4.5: Implementation of the autodual multiplexors

Note that conversely, if such a T_σ exists, the permutation σ is necessarily involutive. This is a consequence of the symmetry condition.

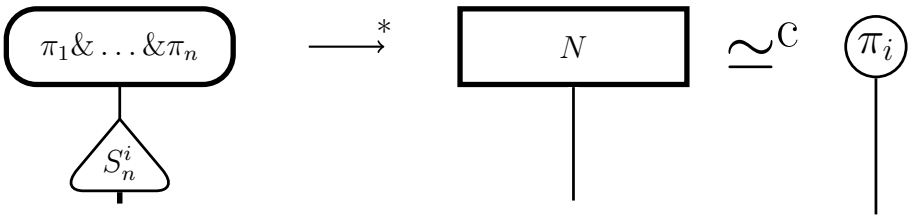
We can see already that transpositors simulate wiring. Since multiport systems are not confluent, it is not enough, as in simple nets, to simulate. We need to check that in some sense, they do not create interference with the environment. Imagine the encoding of a wiring with transpositors, which are only partly reduced into something that is not yet a wiring but that creates some unwished active pairs.

Well, it cannot happen. In fact, the tree structure here is important, but most of all, the fact that auxiliary cells of the encoding of a wiring always stay auxiliary w.r.t. to it. No interreduction is possible between the encoding of the wiring and the environment, so trivially, a wiring and its encoding with transpositors are barbed congruent.

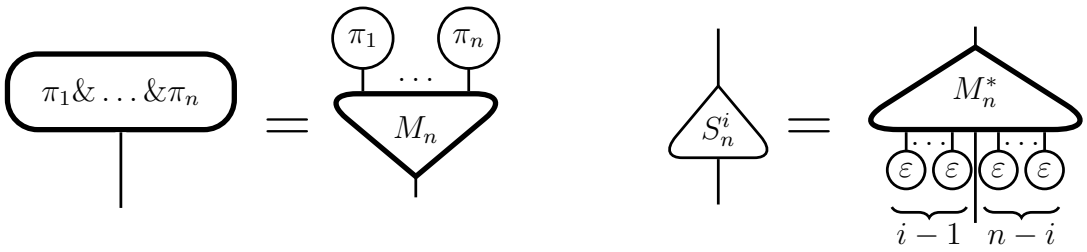
Finally, we use a particular autodual multiplexor which acts like the asymmetric one, but is only available for powers of 2. We only need it for the case with 4 ports:

4.2.3 Menus and selectors

If $\pi_1, \ldots, \pi_n$ are packages, one constructs a new package $\pi_1 \& \ldots \& \pi_n$ (*menu*) which can perform as any of the original nets depending on which of the (reduced) net $S_n^1, \ldots, S_n^n$ (*selectors*) is connected to it:



Here is a possible implementation:



This implementation is good for encoding. With it, the reduction to a net congruent to package π_i is necessary: $M = \pi_1\& \ldots \pi_n \mid S_n^i$ has its unique interface port auxiliary. Since each π_i is a package, any reduction of M is in fact an interreduction of $\pi_1\& \ldots \pi_n$ and S_n^i, which means that M necessarily reduces to a net N composed of the package π_i and some closed nets.

We can do a little better in some circumstances. The following lemma is immediate.

Lemma 4.2.1. *Let ω be a wiring on $a_1, \ldots, a_n$. Then $\varepsilon(a_1), \ldots, \varepsilon(a_n) \to^* \emptyset$.*

Using this, we also show that:

Lemma 4.2.2. *Let $\pi(a)$ be a package with no α-cells. Then $\pi(a) \mid \varepsilon(a) \to^* \emptyset$* necessarily.

Proof. The proof shows a stronger result. Let $\pi(a_1, \ldots, a_n)$ be a reduced net with no α-cells, of which at least one of $a_1, \ldots, a_n$ is a principal port, then $\pi(a_1, \ldots, a_n) \mid \varepsilon(a_1) \mid \ldots \mid \varepsilon(a_n) \to^* \emptyset$. It goes by induction on the size of π. If π is composed of one cell c of power n, it means that all ports of c, except k of its principal ports for some k, are connected 2 by 2. The interaction with ε-cell on the free ports of that net lead to $(n-1)+(k-1)$ ε-cells connected 2-by-2, that reduces to the empty net.

If the package is not a simple cell, then it has a multitree decomposition in which each multitree has non-empty root interface. Pick one of the roots c and reduce any of the $\varepsilon \bowtie c$. The reduction leads to a smaller reduced net with ε-cells connected on all of its interface. If at least one port of the interface is principal, the result is obtained by induction hypothesis. Otherwise the considered net is a wiring, and we use the previous lemma to conclude. □

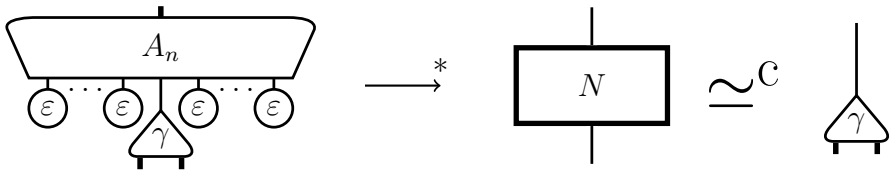
What happens in the presence of α-cells? We can have a multitree $T \equiv \alpha(a, b; x) \mid T'$ in which b is part of the leaf interface, thus bound, and T' is a multitree of root interface x. Then $T \mid \varepsilon(a)$ reduces to $T'\{b/x\}$ which is a deadlock since b is bound.

Lemma 4.2.3. *Let N be a net and $N \to N'$ using a rule on an active pair containing an ε-cell. Then N' has an active pair implies N has a cut.*

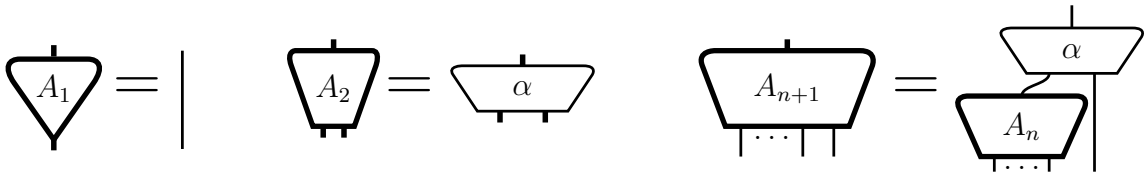
Proof. The only rule with ε that creates wires between ports is the $\varepsilon \bowtie \alpha$. The disappearance of α creating an active pair means that the principal port of α on which the ε-cell interacts was already connected to a principal port of a cell. □

4.2.4 Allocator

When a γ-cell competes with an ϵ-cell on the principal ports of an α, it is always the γ-cell that is finally connected to α's auxiliary cell. We construct, for each $n \in \mathbb{N}$, a net A_n which is in a generalization of that:



The implementation we choose uses α-cells :



4.3 Encoding (restricted case)

With the constructions we gave, we can already simulate multiport interaction nets which are *recursion-free*. Let us consider $\mathcal{S}$ to be a multiport system with alphabet of labels $\theta^1, \ldots, \theta^N$ of respective coarities p_i and arities n_i.

Full, transparent, recursion-free system Let us start with the case in which $\mathcal{S}$ is full and transparent. As the reader might remember, any net can be split in two reduced net, following any splitting of its interface. We apply, for each $k, l \leq N$, $i \leq p_k$, $j \leq p_l$, Lemma 4.1.12 to the RHS of the rules for $\theta^k_i \bowtie \theta^l_j$ (shown in Figure 4.6).

$\Re^{k:i}_{l:j}$ in some sense represents the future of θ^k if it interacts on its i-th port with the j-th port of an θ^l cell. By symmetry, $\sigma_{k:i,l:j} = \sigma^{-1}_{l:j,k:i}$. Since both $\Re^{k:i}_{l:j}$ and $\Re^{l:j}_{k:i}$ are reduced nets, we can apply Lemma 4.1.9 and consider each of them as a reunion of multitrees on all there interface (considering that the connection with the wiring $\sigma_{k:i,l:j}$ is part of it). Such decompositions are not unique, but we just need to pick one initially. If the rule is for a reflexive symmetric active pair, by Corollary 4.1.10, there is a splitting in which each *half* is *the same* (thus named $\Re^{k:i}_{k:i}$) and $\sigma_{k:i,k:i}$ is involutive.

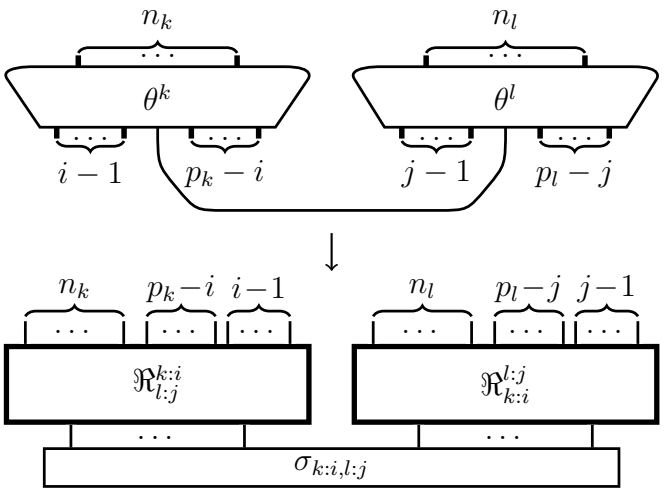


Figure 4.6: Splitting RHS of a rule into two reduced nets.

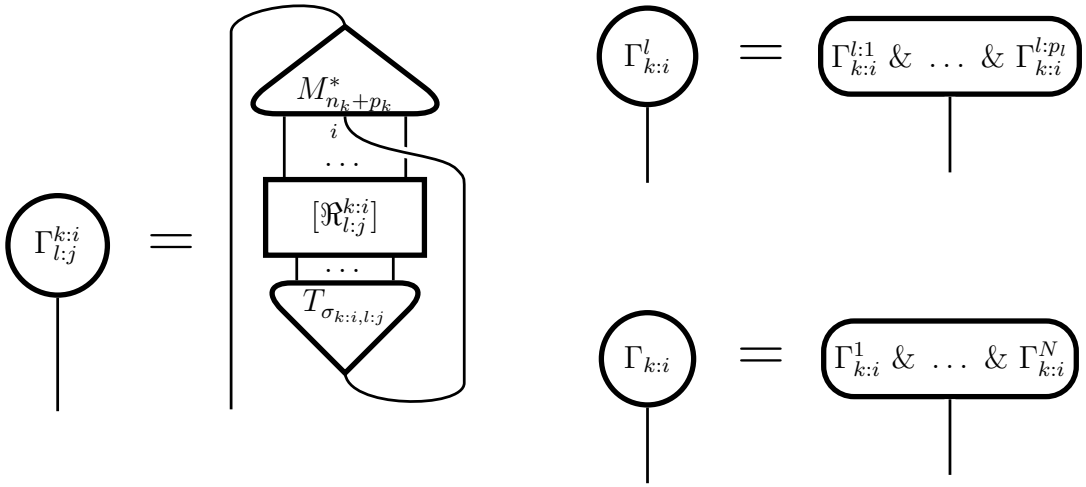
This is important to make sure to encode reflexive symmetric rules correctly, without creating by mistake a multirule.

Definition 4.3.1 (recursion-free systems)**.** Let $\mathcal{S}$ be an interaction net system. It is *recursion-free* if it possible to order its alphabet of labels into $\theta^1, \ldots, \theta^N$ in such a way that $\Re^{k:i}_{l:j}$, when defined, contains only symbols that are strictly smaller than θ^l (and symmetrically $\Re^{l:j}_{k:i}$, when defined, contains only symbols that are strictly smaller than θ^k).

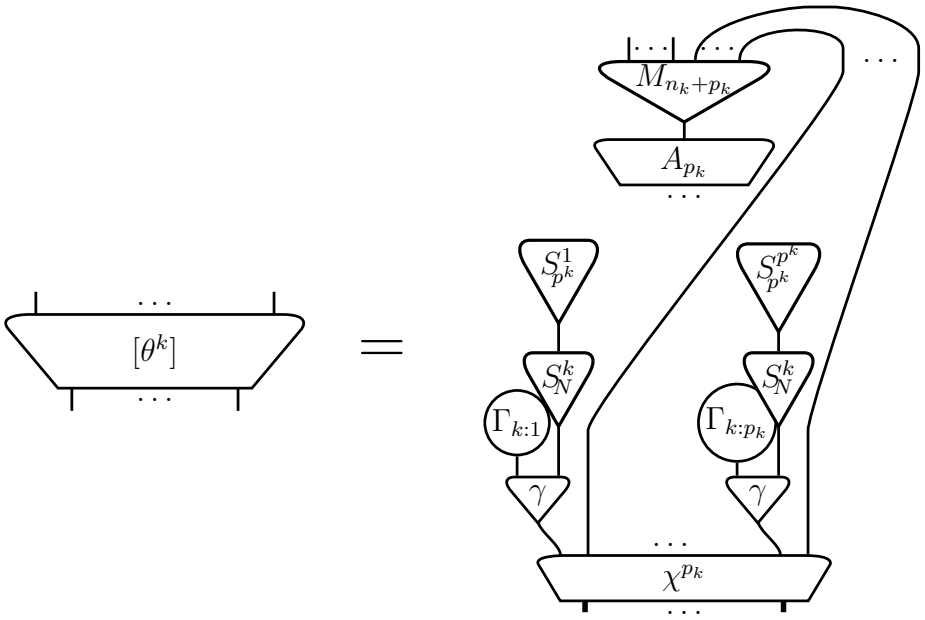
With this order, a reduced net can be defined for each θ^i inductively, encoding in it all of the possible futures of each of its opponents. This is our definition of *recursion-free*.

We construct, for any $k, l \leq N$, $i \leq p_k$ and $j \leq p_l$ the following packages which represent

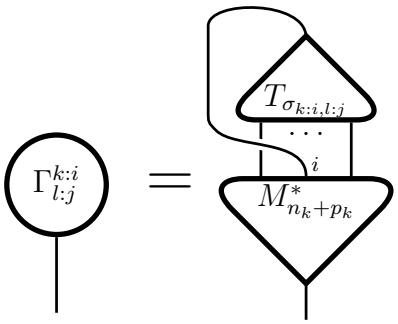
- $\Gamma^{k:i}_{l:j}$: the future of θ^k if interacting on i with the j-th port of θ^l;
- $\Gamma^{l}_{k:i}$: the possible futures of θ^l if interacting with the i-th port of θ^k;
- $\Gamma_{k:i}$: the possible futures of all the cells if interacting with i-th port of θ^k.



$[\Re^{k:i}_{l:j}]$ stands for the net obtained by replacing each cell θ^k by $[\theta^k]$ in $\Re^{k:i}_{l:j}$, where the square bracket for a cell means:



The iteration can start because the possible futures of minimal cells for that order are wirings alone. Let us suppose θ^k is a minimal cell, then:



so $\Gamma^k : i$ can be built for any i and therefore $[\theta^k]$. A simulation of an interaction is given in Figure 4.7.

It is important to notice that, even though multiport systems are not confluent, once an interaction $\chi \bowtie \chi$ has taken place (Fig. 4.7(b)), the rest of the reduction is necessary, to a certain point. From here, things can happen in parallel: erasors can do their erasing job, or the only non erasor cut can be reduced. Once it is reduced, the net is split in two independent parts, each representing the future of each of the cells. The two futures of the cells are shown in parallel in the generic reduction, but in fact, they can each reach there state of Fig. 4.7(f) separately. Until neither of them does, the whole net is independent from the rest of the world: all its interface is composed of auxiliary ports. Once one part has reached its 4.7(f)-state, the encoding can start the simulation of a new active pair, entangled with the first one.

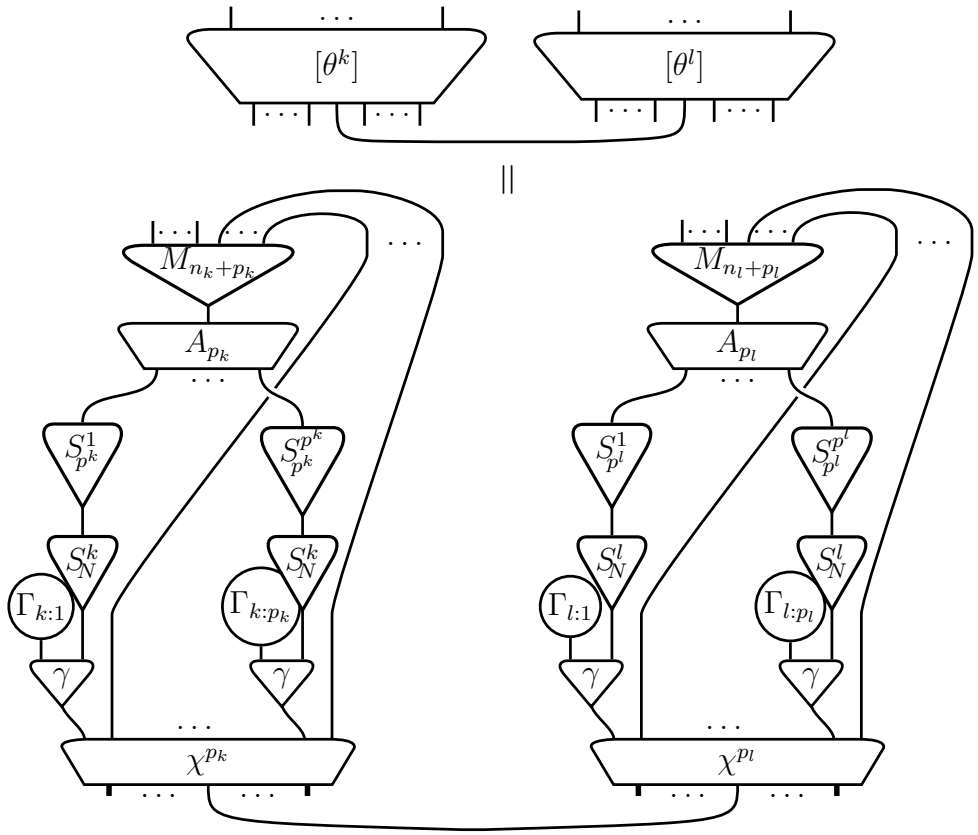
Let $\mathcal{S}$ be a multiport system and $\mathcal{T}$ its encoding into multiport combinators. Then, for any net $M \in \mathcal{S}$, it is clear that the encoding is complete: $M \to N$ implies $[\![M]\!] \to^* M' \simeq^{\mathrm{c}} [\![N]\!]$. We have shown which reduction does that. A precise strategy: reduce an $\chi \bowtie \chi$ active-pair only if there are no other active pairs.

For soundness, we need to notice that each cell of encode M and its reductions can be *tagged* as *part of* a cell of M or of its reductions. Even more, there is a one-to-one correspondence between cells of M and its reductions and χ^p-cells of $[\![M]\!]$ and its reductions. Any reduction $[\![M]\!] \to^* N$ can be *back-simulated* by $M \to^* M'$, reducing in M all cells corresponding to χ^p-cells that have been reduced during $[\![M]\!] \to^* N$. N can have all its other active pairs reduced, leading to $N'' \simeq^{\mathrm{c}} M'$.

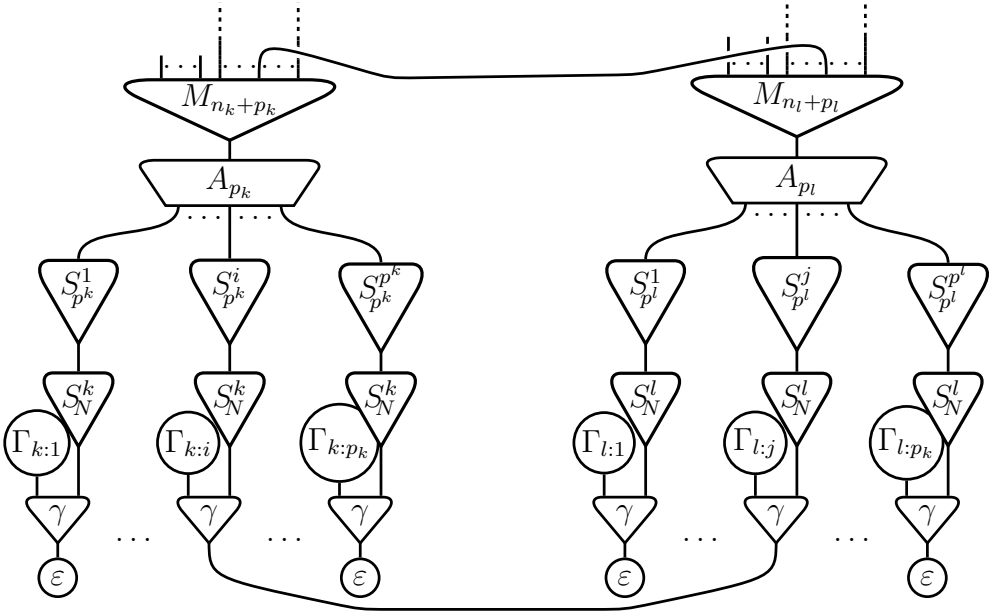
We can see in the generic reduction simulation why the encoding does not reduce to the encoding of a reduction: the packages that were not used by the choice of the reduction are plugged with an erasor which makes sure they have no computational value but which do not get necessarily erased because of the presence of α-cells. It is possible to get rid of any α-cell in those packages, by using then a *decoder* to place back the missing cells. We will see how to do this when we will get rid of δ-cells as it is primordial there. Applying this technique to α-cells does not bring much explanatory power to the encoding, just a better garbage collector system.

More interesting, the *determinism* of the allocator A_p suggests it is possible to avoid it completely, as the reduction of the χ-cell already determines its issue. It is in fact possible to modify χ for a more complicated rule which encompasses the allocator. The rule is rather unnatural. Since we are looking for minimal actions of concurrent processes, we believe an allocator cell is not misplaced. We would need a cell of coarity 2 anyway for the blocking mechanism.

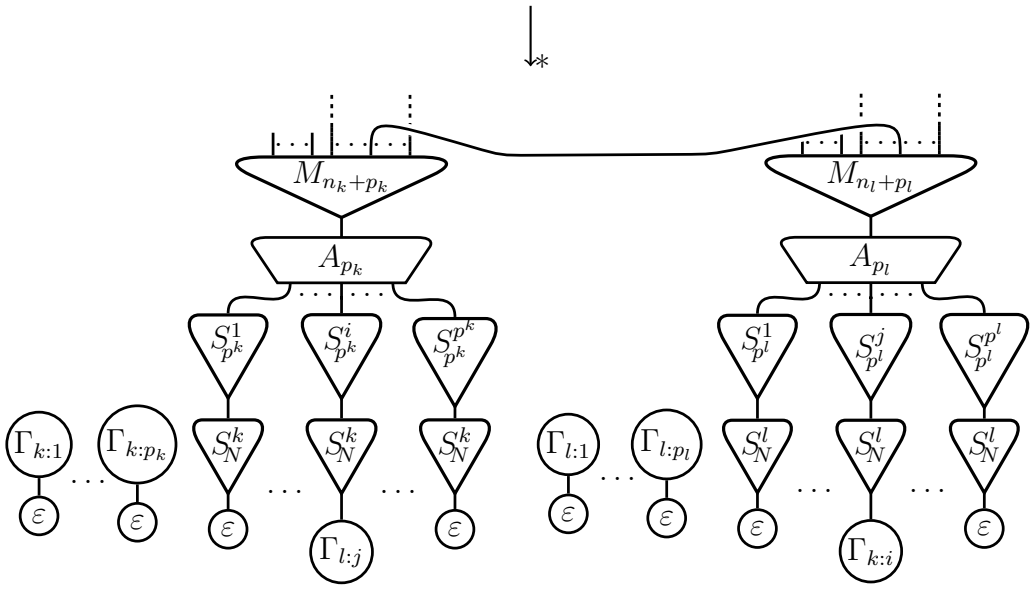
Adding opacity In the paragraph above, we have not taken good care of observability. We considered any *start* of simulation to be observable, since it would eventually end-up in the encoding of the reduction. This is good enough if all rules of $\mathcal{S}$ are observable. But how to handle unobservable rules? In the presence of those, it can-



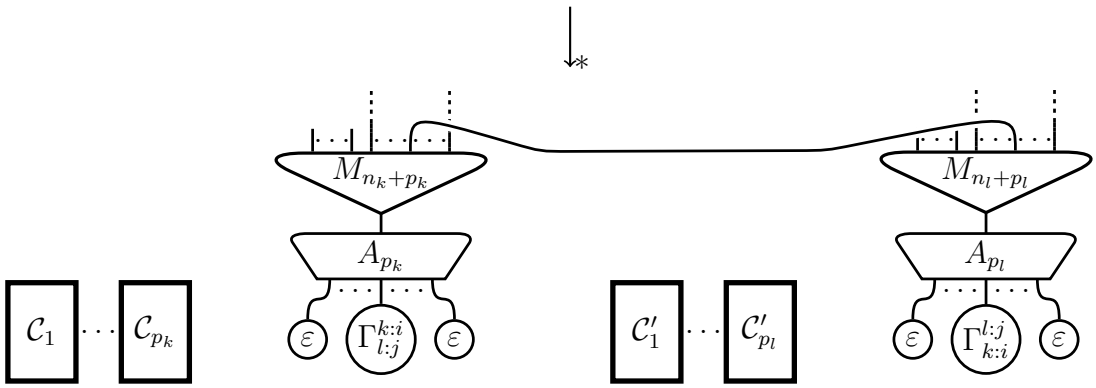
(a) An active pair in the encoding of a net



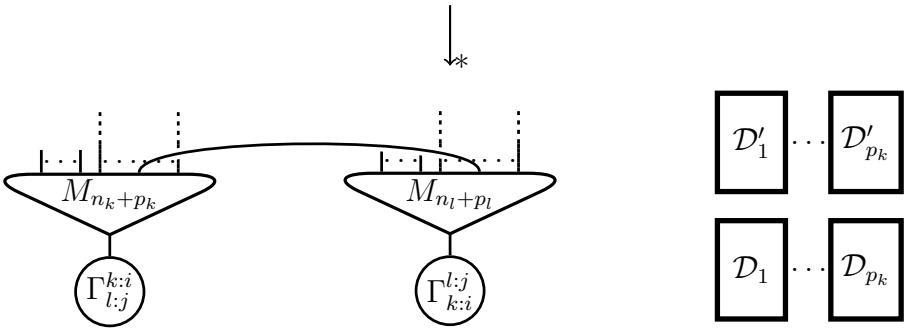
(b) First step: the simulation is on.



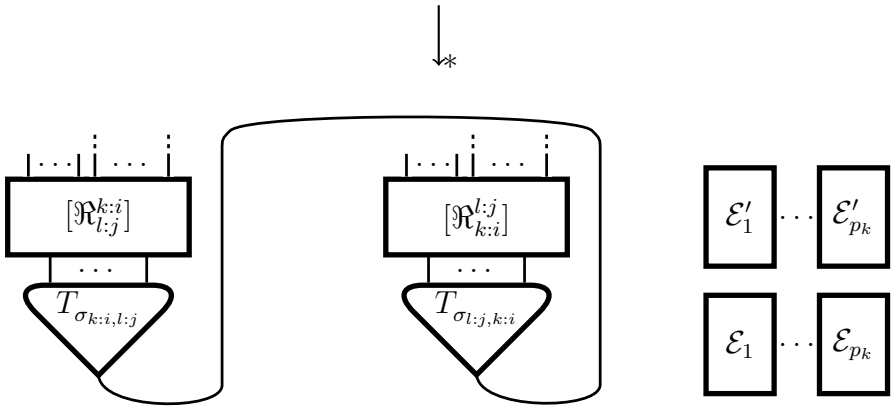
(c) Exchange of possible futures.



(d) Each *cell* extracts its correct future.



(e) No choice here.



(f) All is left to simulate is the wiring

Figure 4.7: Simulation of the interaction for $\alpha^k \bowtie \alpha^l$

not be a characteristic of some $\chi^k \bowtie \chi^l$ active pairs to be observable or not, since it could happen that the active pairs employ cells of same coarities, but one is observable and the other not. What is true of χ/χ-rules, is true for any other rule used in the simulation above. We therefore need to introduce a new rule which is observable.

As the reader might notice, we will later introduce even more rules (from the ones given in the beginning of the section). Maybe one of those could be the observable equivalent of the encoded rule. We assure him that it is not the case, as he will understand from the following paragraphs. It is not possible to use the decoding mechanism as observable, since it might be produced at several places at the same time leading to a surplus of observability.

Let us proceed. The system is very simple. If a rule is observable in $\mathcal{S}$ and we want a unique reduction step to be observable in $\mathcal{T}$, we need it to be applied between cells from both halves of the reduction simulation.

By a quick overview of the simulation, we can see no interaction can happen between both halves before reaching the stage of simulating the wire (Fig. 4.7(f)). It is sufficient then to add an observable but otherwise useless cell at the root of each of the trees $T_{\sigma_{k:i,l:j}}$ and $T_{\sigma_{l:j,k:i}}$. The only change to the whole encoding is that if, and only if, $\theta^k_i \bowtie \theta^l_j$ is observable in $\mathcal{S}$, the package $\Gamma^{k:i}_{l:j}$ uses an extra o-cells, as shown in Figure 4.8.

Inactive pairs We have considered until now that any cut-like configuration between two cells in $\mathcal{S}$ has a rule. In such a case, any χ^k/χ^l pair in a net $[\![N]\!]$ can trigger the chain of interactions, since it necessarily simulates some reduction of the original net. It is not the case that all systems are full. Actually, the separation principal auxiliary ports is ad-hoc in a multiport setting, and is used only for clarity. If we consider the *a posteriori* definition of principal ports, we need to be able to encode the fact that some connections between ports do not define active cells.

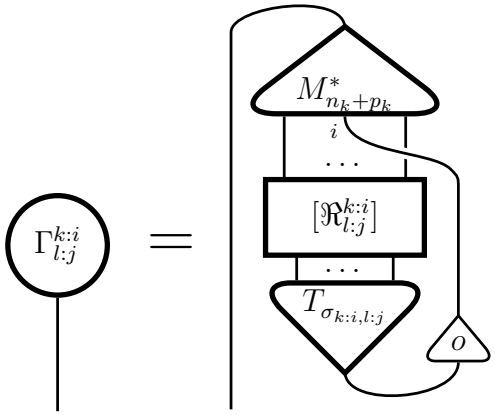
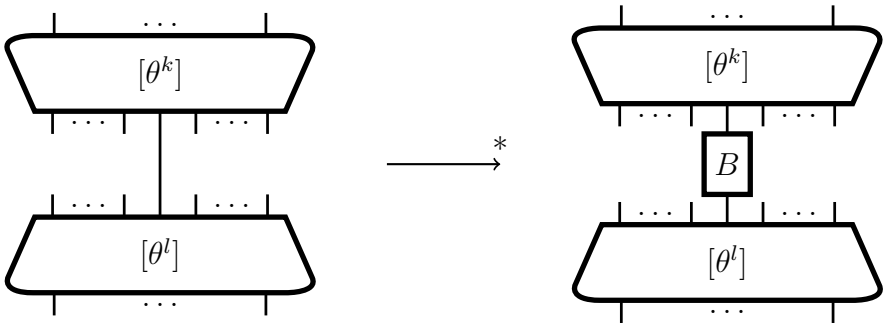


Figure 4.8: Possible future if the interaction is observable.

In simple nets, a cut between two principal ports defined necessarily a deadlock. It was easy then to consider the encoding of $\theta \triangleleft \triangleright \theta'$ to be any deadlock that is splittable in half and use it as the reduced of $[\![\theta]\!] \mid [\![\theta]\!]$. It is not true anymore in a multiport setting. In fact, it is possible for instance, to have a rule for $\theta_1(a,b,c) \bowtie \theta'_1(a,d)$, making the first port of θ principal, and no rule for $\theta_1(a,b,c) \bowtie \theta_1(a,d,e)$, making a cut-like configuration an inactive pair.

A simple approach is to consider then $\theta \triangleleft \triangleright \theta'$ to be the active pair $\theta \bowtie \theta'$ itself. We would then simply consider $\Re^{\theta'}_{\theta}$ to be θ, for any θ, θ' in the language, and of course make that rule unobservable. This is satisfying if one does not consider divergence as irritating. The encoding of a net containing such an active pair could go on for ever just trying to simulate it and go back to itself.

In order to avoid this cyclic reduction, we introduce a blocking mechanism. When an inactive pair is simulated, it reduces to itself, except that the wire that created the active pair now contains a blocker cell, pointing toward both principal ports, thus the need of a cell of coarity 2. If one of the involved cells is later reduced by another of its ports, its simulation sends unblocking cells on all the ports of its interface. In this way, the cell that has not been modified gets its use of a principal port restored, in case it needs it.

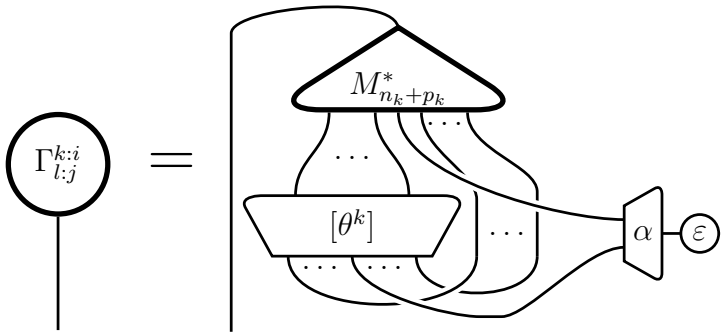


If $k : i \neq l : j$, it is sufficient to attach the net B to one of the halves of the *almost* active pair. The problem is is $k = l$ and $i = j$, in oder for the encoding to be defined

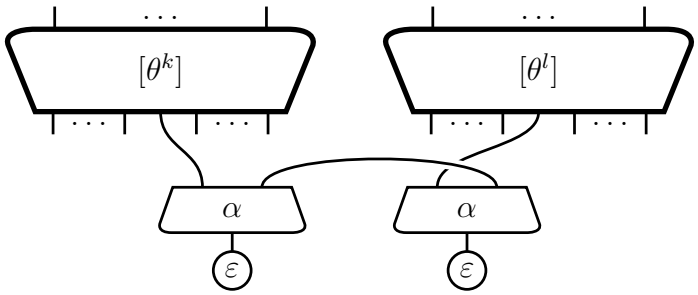
correctly, we need for B to be created from the connection of two smaller nets B. It is not a problem. We consider

$$\mathrm{B}(x, y) = \alpha(x, y; z) \mid \varepsilon(z).$$

For any (k, i, l, j) which has no rule, we define $\Gamma^{k:i}_{l:j}$ the following way:

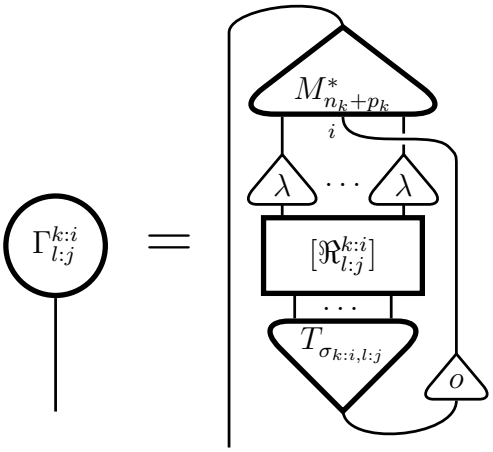


In this way, the *end* of the simulation contains, instead of $[\theta^k] \mid [\theta^l]$, a net:



which has no active pairs, since we explicitly stated that $\alpha \bowtie \chi^p$ has no rule.

On the other hand, for any (k, i, l, j) which has a rule in $\mathcal{S}$, we define $\Gamma^{k:i}_{l:j}$ the following way (remove the o-cell if the original rule is not observable):



In this way, if the net representing θ^k in the blocked situation has interacted by another port simulating an actual reduction rule, a λ-cell is sent towards the α-cells, erasing them one after the other, and vanishing against the χ^l-cell of the other part of the blocked net. If a unblocker cell has been also *sent* by the other side, the two λ-cell vanish when they encounter, creating the wished connection between the wished ports.

If λ-cells are sent toward non-blocked wires, they mean no harm and just get erased by contact with the χ-cell, if it exists, or as soon as it starts existing.

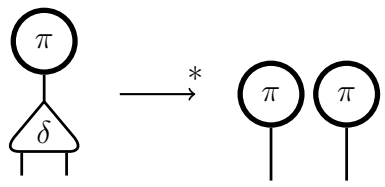
4.4 Encoding systems with recursion

We have seen in the previous section how to encode any multiport system that has no recursion (see Definition 4.3.1). We have seen how to deal correctly with observability, and inactive pairs. In this section, we show how to encode systems with recursion, which is the main accomplishment of Lafont in [33]. The principle is exactly the same, so if the reader is already aware of Lafont's work, he will learn nothing here. We only adapt it for a world with multiport cells.

Lafont's intuition was that it is possible to create a big big package containing all possible futures of all cells (we add: interacting on all possible principal ports). Every cell-encoding has access to a copy of that DNA-package and takes in it only what it needs. To do so, it first copies it for personal use as many times as it needs (and always keeps a full-copy for further duplication). Each part inside the net extracts the needed piece from a copy that is given to it. The system is able to perform, in some sense, *self-duplication* and *specialization*, thus the DNA metaphor.

4.4.1 Duplication

We can see how the whole system is based on duplication and therefore on a strong use of the δ-cell. Unfortunately, δ does not copy itself. Nevertheless, a package π without δ-cells can be duplicated as follows:

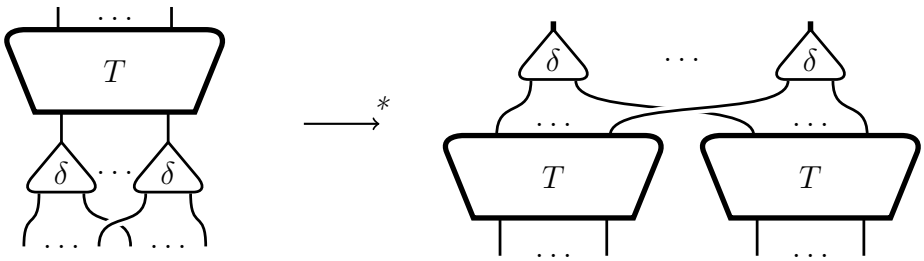


It is a corollary of the following lemmas, and the decomposition of reduced nets (Cor. 4.1.11):

Lemma 4.4.1 (duplication)**.**

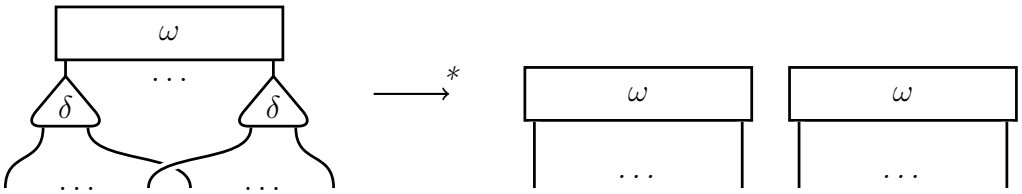
1. *For any multitree*[2] $T(\tilde{a}, \tilde{x})$ *without* δ*-cells, of root interface* $\tilde{a} = a_1, \ldots, a_p$ *and leaf interface* $\tilde{x} = x_1, \ldots, x_q$*:*

$$T(\tilde{a}, \tilde{x}) \mid \delta(a_1; b_1, b_1') \mid \ldots \mid \delta(a_p; b_p, b_p') \to^*$$
$$T(b_1, \ldots, b_p, y_1, \ldots, y_q) \mid T(b_1', \ldots, b_p', y_1', \ldots, y_q') \mid \delta(x_1; y_1, y_1') \mid \ldots \mid \delta(x_q; y_q, y_q')$$



2. *For any wiring* $w(a_1, \ldots, a_n)$

$$w(\tilde{a}) \mid \delta(a_1; b_1, b_1') \mid \ldots \mid \delta(a_n; b_n, b_n') \to^* w(b_1, \ldots, b_n) \mid w(b_1', \ldots, b_n').$$

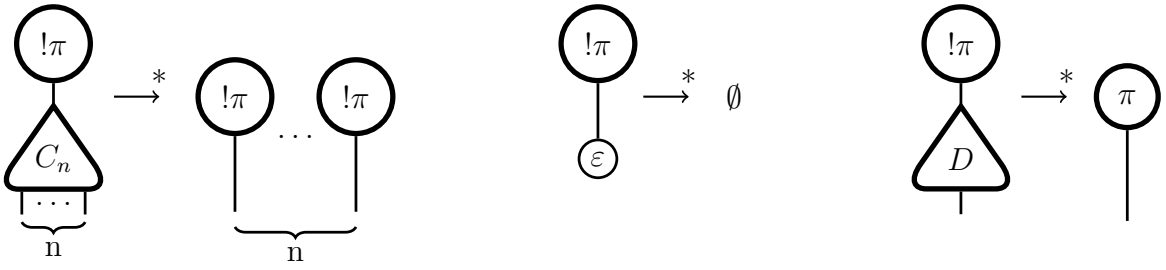


Both statements are proved by induction on the size of the tree or the wiring. The rule for the active pair $\delta \bowtie \delta$ is primordial for the correct duplication of wirings. It is therefore not possible to change its rule to have δ duplicate itself.

4.4.2 Codes, copiers and decoder

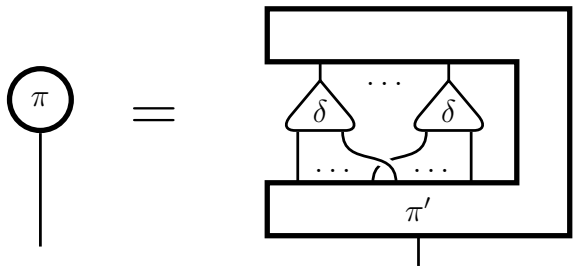
One constructs, for any package π another package $!\pi$ (*code* of π) which can be duplicated, erased and from which π can be extracted. More precisely, one constructs reduced nets C_n (*copier*) and D (*decoder*) with the following properties:

[2]We remind that not all ports of the leaf interface are connected to auxiliary ports even though they are represented as such (see Def. 4.1.7).

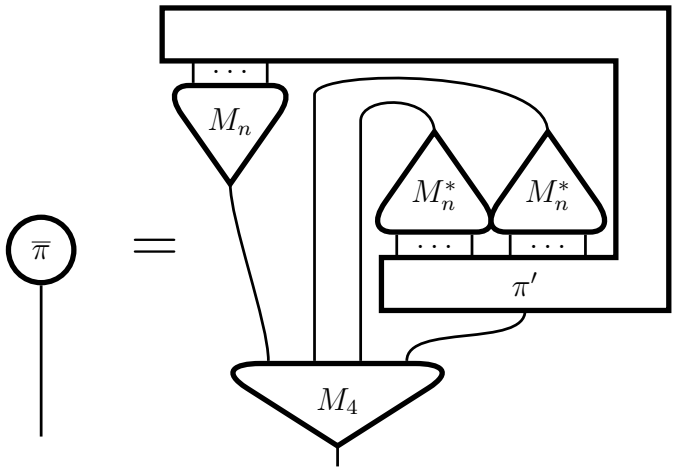


The reader can notice the use of tree to graphically represent C_n and D. It is just because the implementations we (meaning Lafont) propose are in fact trees.

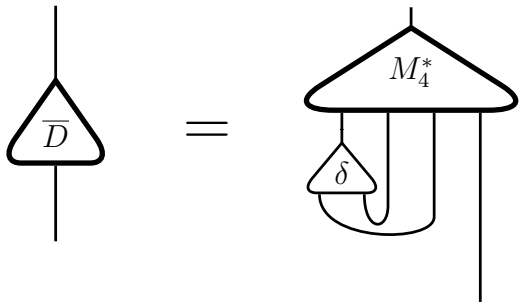
We start by dealing with duplication, building for π a net $\overline{\pi}$ that can be duplicated and decoded. If a package really has no δ-cell, a simple implementation of C_n is T_n: a tree of δ-cells. Any tree of these cells with enough auxiliary ports does the job as well. We define $\overline{\pi}$ and $\overline{D}$. The main remark is that if π contains n δ-cells, it can be decomposed as follows:



where π' is reduced with $3n+1$ fee ports, and contains no δ-cells. An implementation for $\overline{\pi}$ can therefore be:



In which case, an easy way to fill in δ-cells back where they belong is the tree $\overline{D}$:



Here again, since all cell involved are simple cells, there is a kind of partial confluence. Anyway, we just use the fact that

$$\overline{\pi}(a) \mid \overline{D}(a,b) \simeq^{\mathrm{c}} \pi(b).$$

We got rid of δ-cells in order to create some duplicable packages. We can use the same strategy to create erasable packages. The only problematic cell for erasure is α. When erased using ε, its other two ports are not transmitted erasers, situation that can end with the creation of a deadlock. A simple way of assuring that a package gets erased by ε is for it not to have α-cells. This is done by a very similar construction, given in Figure 4.9. We call $\underline{\pi}$ the package π coded as to not contain α-cells. $\underline{D}$ is the generic name for the tree allowing to place back in $\underline{\pi}$ the α-cells in order to obtain π back. These two constructions use autodual multiplexors instead of γ-cells, so *this* coding has to be done before the one getting rid of δ-cells.

We define $!\pi$ to be the result of removing δ-cells from $\underline{\pi}$ (if we went the other way, we would have added δ-cells after getting rid of them). Finally, $D(a;b) \equiv \overline{D}(a;x) \mid \underline{D}(x;b)$.

4.4.3 The encoding, general case

A big duplicable package containing all possible futures of all possible cells for all possible interactions is constructed. It is simply called Γ. It can be erased and duplicated. For each cell θ^k of $\mathcal{S}$, of arity m and coarity p, we construct the net denoted $[\theta^k]$, containing this package Γ and decoders D. This net is capable of retransforming Γ into the "real" menu of all possible futures and extract from it each piece that is necessary. It is shown in Figure 4.10.

It is not possible to build the big package Γ with the help of these nets, since they contain Γ. Instead, for each cell labeled θ^k (of arity m and coarity p) is given another net, denoted $< \theta^k >$ with an interface composed of $p+1$ principal ports and m auxiliary ones. This cell is like a squeleton of $[\theta^k]$: it contains the decoders and selectors, but does not contain Γ directly. When Γ is connected to it, at the right place, it "transforms" itself into $[\theta^k]$ by first duplicating Γ to keep a full copy and then decoding and selecting the right pieces.

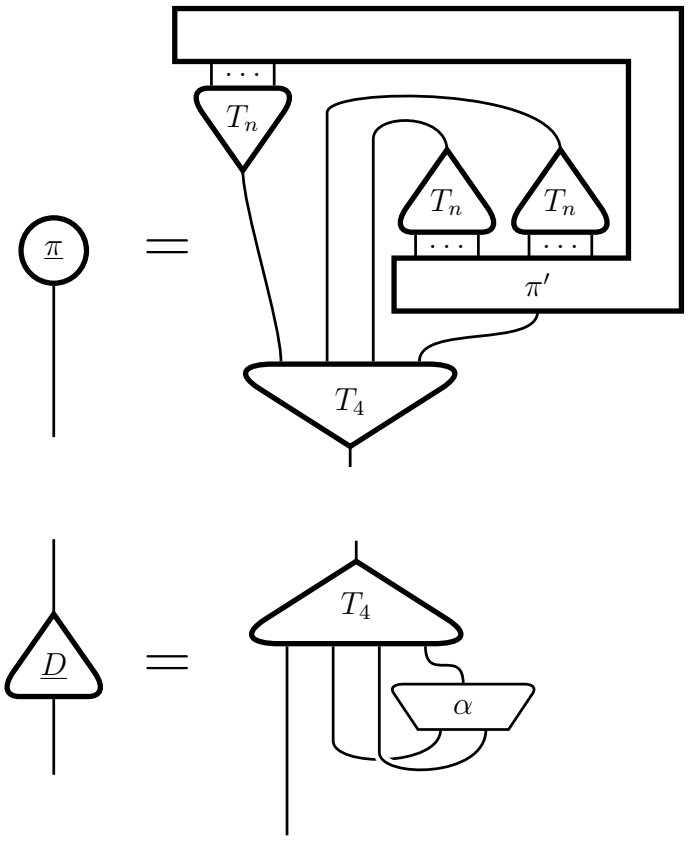


Figure 4.9: Coder and decoder to get rid of α-cells.

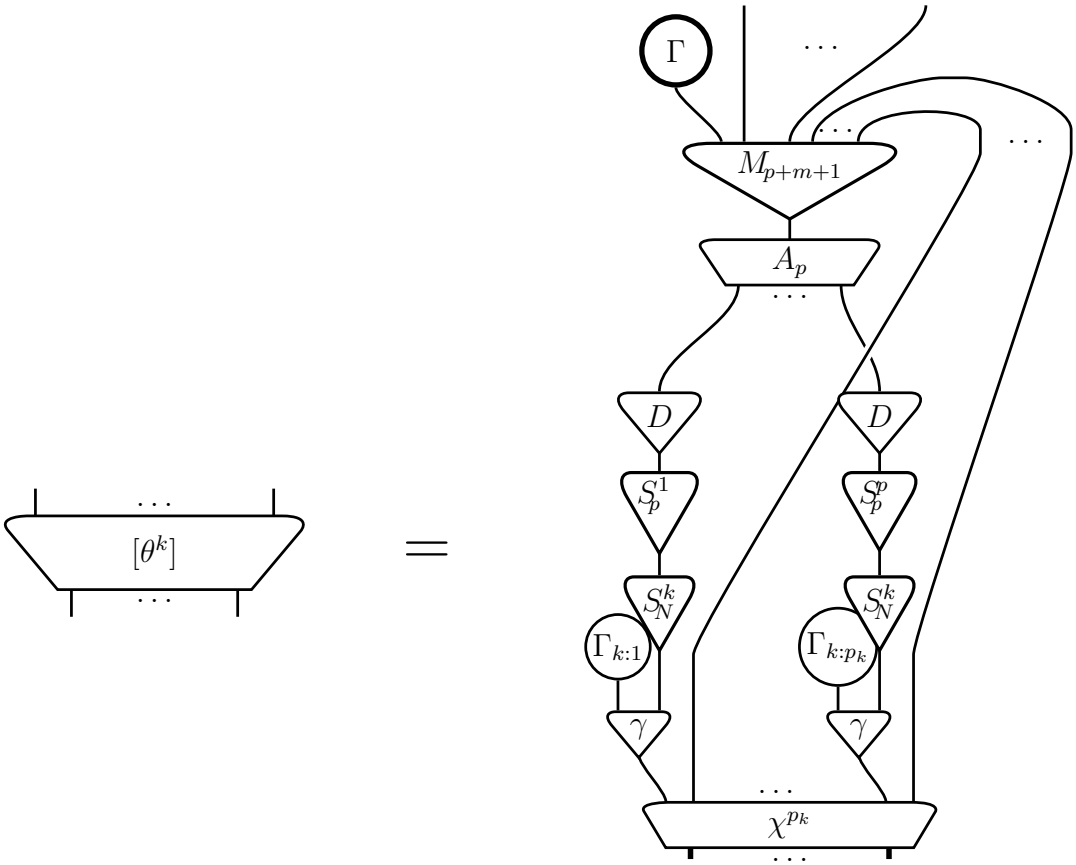


Figure 4.10: Encoding of a cell θ^k of coarity p and arity m.

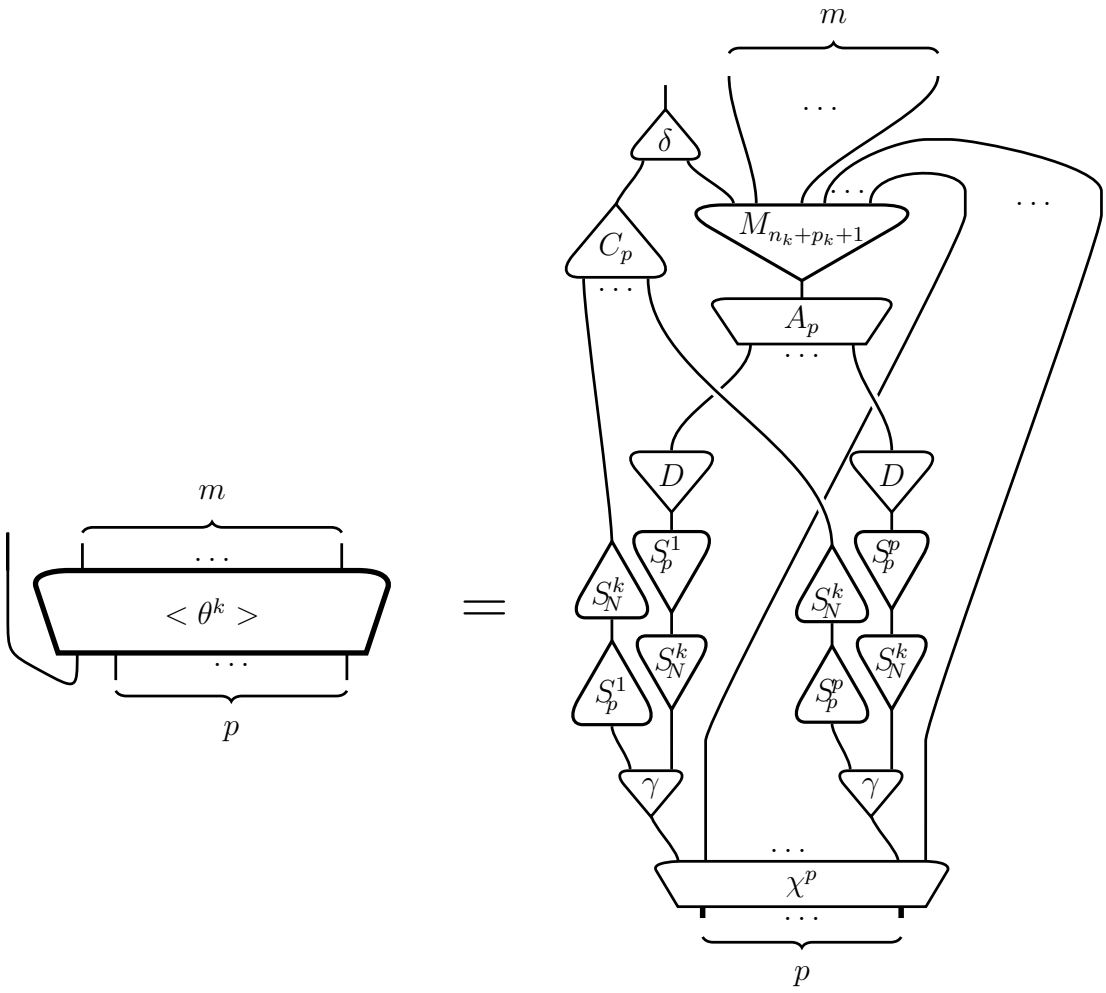
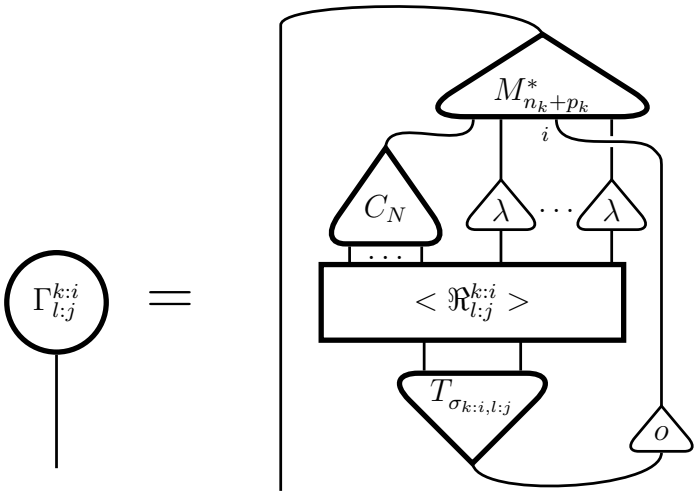


Figure 4.11: The net that is used instead of $[\theta^k]$ to build the packages of possible futures.

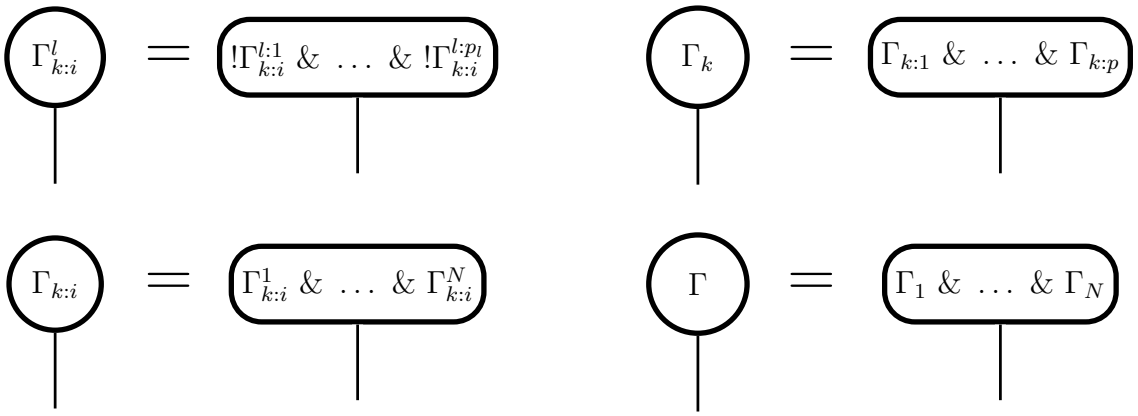
The way of doing so, is of course by using copiers and selectors (see Figure 4.11). We can see that it copies Γ as many times as necessary (one for each principal port) and once more, this last copy being connected as auxiliary package for further use.

To encode the halves of right-hand-sides of rules, we build a net that acts like an empty receptacle for a real encoding of a cell, like a mold. It contains a net $< \theta^k >$ for every cell of arity k that the corresponding half-rule contains and copiers that send a copy of Γ to each of them. With these molds, we build the packages containing the possible futures of a cell. Let N be the total number of cell of $\Re_{j:l}^{k:i}$.



The copier C_N is in fact connected to *each and every* net $< \theta >$ in $< \Re_{j:l}^{k:i} >$, on the principal port of the δ-cell at the top.

Let us now build the full genetic code Γ. If we follow the wished sequence for the simulation, we can see that Γ will first be copied, thus needs to not have δ-cell. This is why $\Gamma_{k:i}^{l}$ is now defined with the help of $!\Gamma_{k:i}^{l:1}$ packages. Then packages $\Gamma_{k:i}$ get extracted for each $1 \leq i \leq p$, once k has been determined. Let p_l be the coarity of θ_l.



Since $!\Gamma_{k:i}^{l:j}$ has no δ-cell, none of the packages above do either (see Definition 4.2.3). In $< \theta^k >$, a copy of Γ is performed for each of θ^k's principal port. Out of this copy

is first extracted a package Γ_k. If one is interested in performance, it is possible to exchange these two operations, and extract Γ_k before copying. Out of each Γ_k, each port gets its corresponding $\Gamma_{k:i}$. Then, the interaction is simulated as before: the opponent port which *wins* the interaction extracts $\Gamma^l_{k:i}$ and $\Gamma^{l:j}_{k:i}$ for him to use, obtaining its future, which is injected with the needed δ and α-cells at the right places by the use of D-trees.

4.4.4 Correctness of the encoding

We denote by $[\![\_]\!]$ the encoding described along this entire section, which is, for any $\mathcal{S} = (|\mathcal{S}|, \mathcal{R}, \mathcal{O})$, a function from $|\mathcal{S}|$ to $|\mathbb{MC}|$ describes, for each label θ of $\mathcal{S}$ by $< \theta >$.

Thanks to our use of coding nets into δ and α-free nets, the previously floating-around-the-main-computation-closed-nets are now composed of a package with no α-cells, erased by a ε, making them totally erasable. This way, we get the following theorem:

Theorem 4.4.2 (Strong completeness)**.** *For any multiport system $\mathcal{S}$, $[\![\_]\!]$ is such that for any net M from $\mathcal{S}$, $M \to^* N$ implies $[\![M]\!] \to^* [\![N]\!]$.*

Proof. It is sufficient, for each interaction in the reduction of M to N, to follow the perfect simulation that leads to the correct encoding. □

Conversely:

Theorem 4.4.3 (Soundness)**.** *For any multiport system $\mathcal{S}$, $[\![\_]\!]$ is such that for any net M from $\mathcal{S}$, if $[\![M]\!] \to^* M'$, then there are nets $N \in \mathcal{S}$ and $N' \in \mathbb{MC}$ such that*

$$M \to^* N \qquad \textit{and} \qquad M' \to^* N' \simeq^{\mathrm{c}} [\![N]\!].$$

In fact, what we imagine N' to be is not exactly $[\![N]\!]$ because of the blocking mechanism. When M – or one of its reductions – contains an irreducible cut, its encoding can start to simulate it before to enter in the blocked process. Then, N' is $[\![N]\!]$ in which the wire connecting the inactive pair is replaced by a *blocked wire*. This is the only case in which soundness if up-to barbed congruence. Therefore we have the following theorem:

Theorem 4.4.4 (Strong soundness)**.** *For any* full *multiport system $\mathcal{S}$, $[\![\_]\!]$ considered without the blocking mechanism is such that for any net M from $\mathcal{S}$, if $[\![M]\!] \to^* M'$, then there is a net $N \in \mathcal{S}$ such that*

$$M \to^* N \qquad \textit{and} \qquad M' \to^* [\![N]\!].$$

4.5 Quality of the combinators

The encoding fits Gorla's criteria [26], which speak about the quality in terms of computability comparison. We would like to make a few comments on the system of combinators itself. The search for combinators for a language, or set of languages, has two purposes.

A practical one first. It is particularly true for interaction nets. These are a framework, more than a language, and allow a great deal of freedom. Cells can represents information at any scale, from operations in machine language to computers on a network. It is maybe not the best language to study each of these scales, but its universality can hardly be denied. Having a system of combinators puts a little bit of order in that chaos. It gives a natural scale for interaction nets. In fact, it gives a natural interaction net language, if we wish for instance to implement a programming language based on interaction nets.

The second purpose, which was the one driving Lafont, was theoretical. The question is: what are the real operations behind a calculation? Several answers can be given. We know that behind arithmetics stands, as most important, the operation "add 1". Hard to get smaller that that. Sequential computation can be modeled by Turing machines, in which the basic operations are: read, write and move right/left. But it can also be modeled by Λ-calculus, which has one rule, necessarily the minimal one we are looking for: substitution. Can substitution be considered minimal? As Lafont remarks, a more primitive system is combinatory logic, with two rewriting systems: erasing and duplication. Most probably, there is no real answer to the minimality problem, as it is most likely to be *cyclic* in some manner. Think of a dictionary. Which are the minimal words needed to start the definition process? Our purpose is to give one possible answer to the minimality question for concurrent computation.

A quality system is one that has a good balance of simplicity throughout all its components. Interaction nets have two main components: the alphabet and the rules. A good balance of simplicity is one that has as little different cells as possible, but not at the price of complicated rules. For instance, interaction combinators for Lafont nets have three cells and six rules. Bechet on the other hand [4] shows how to combine several cells into one, obtaining, from Lafont's combinators a universal system with only two cells. (The ε erasing cell cannot be combined with others). The system is universal, but the *interesting* rule is so complicated, it has no meaning to us. We tried to follow this idea of simplicity over minimality. We now discuss a little bit some of the ways the system could loose cells and rules.

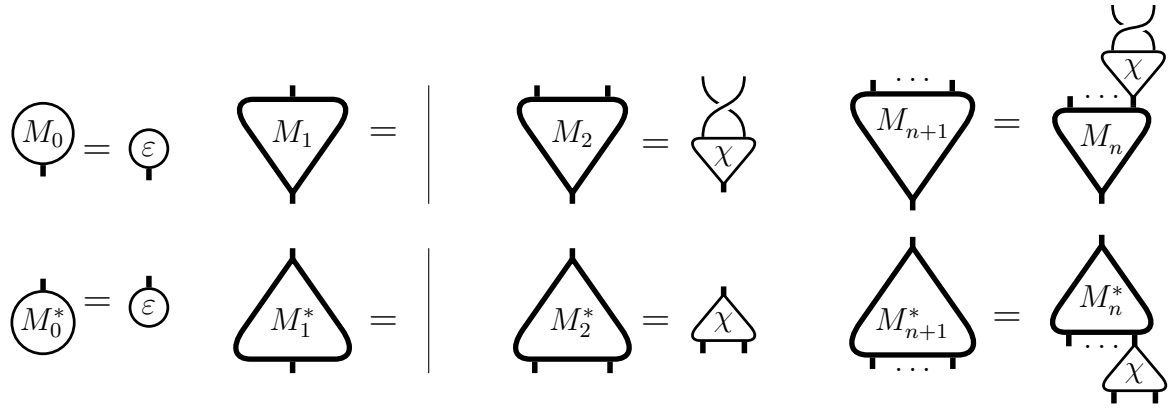
Unlike Lafont, we chose to distinguish the duplication cell from the cell representing the root of *encoded cells*. We can see anyhow that $\chi^1 \bowtie \chi^1$ has the same rule as $\delta \bowtie \delta$, so χ-cells can be considered as generalized δ. It is possible to consider so, but in this case χ-cells cannot be duplicated. The code mechanism that gets rid of δ-cells needs to get rid of each χ^n cell that can be encountered in the system. Luckily, given a system

$\mathcal{S}$, this quantity is finite. It just seems that the *choice* mechanism and the *duplication* mechanism are not the same and combining them in this way is more cheating then getting a new insight on what minimal operations are.

Another way of dealing with that is to consider γ-cells to be the duplicator. And in fact, it is possible to code any package without γ-cells and build a decoder, using autodual multiplexors instead of normal ones, which means duplication can take place. Menus and selectors can also be built with autodual multiplexors. Finally, we would use χ^1 as the winner of the allocator cell α, since the package $\Gamma^{l:j}_{k:i}$ would be built with autodual multiplexors too, using only δ-cells. A complication of using γ as a duplicator is that it is hard to follow the rule, since each port of a duplicated cell has to be in some sense connected to the *wrong* auxiliary port of γ in order for the process to continue without interleaving duplications of different parts of nets. It is, nevertheless, a not that hard way of reducing the number of cells.

$$\beta(a_1,\ldots,a_n) \mid \gamma(a_i;x,y) \rightarrow$$
$$\beta(b_1,\ldots,b_n) \mid \beta(c_1,\ldots,c_n) \mid [b_i,x] \mid [c_i,y] \mid \gamma(a_1;c_1,b_1) \mid \ldots \mid \gamma(a_n;c_n,b_n)$$

The only place autodual multiplexors cannot do the job alone is to create transpositors. It is possible anyhow to create multiplexors with χ^1-cells only:



Of course, this means we need to get rid of γ-cells for making packages duplicable.

Another, more profound remark, is the one about the use of allocators. Even though the rule is quite clear and is natural in computer science (think of allocating resource), it is not necessary in combinators. A χ/χ interaction already defined who has to *win* the allocator: all of its other ports receive ε-cells. It is therefore possible to modify χ-cells to integrate the allocator: each χ^n-cell would have an extra auxiliary port x which would correspond to the auxiliary port of A_n. The interaction of χ/χ would directly connect the correct auxiliary port of the decoder with this port x. This mean the tail of the decoder D has to be an auxiliary port of χ. Then, for each principal port χ has 3 auxiliary ports, plus the one, therefore $\mathtt{ar}(\chi^n) = 3n+1$. We do not give the detail of the encoding, but one possible rule is given in Figure 4.12.

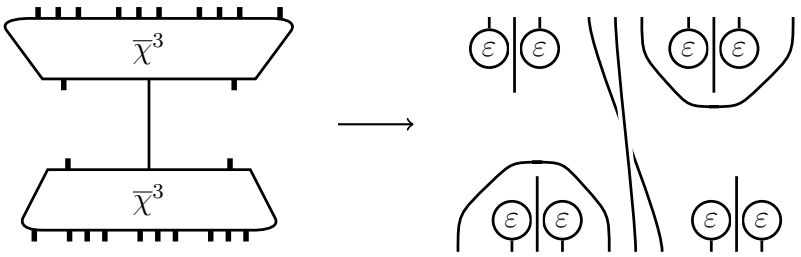


Figure 4.12: A possible rule that combines χ_n-cells and allocators choice.

Such a rule seems a little bit ad-hoc for our purpose. It does not bear at all the simplicity we would wish from a universally useful cell. Moreover, it would still not dispense us from the use of a 2-multicell for the blocking mechanism. In fact, if we get rid of α, the only cell with multiple principal ports becomes the χ-family. A careful analysis of the blocking mechanism shows that a cell with at least two principal ports is needed, but all cells in the χ-family need to interact with each other. We do not win much by complexifying the χ-cells to contain the allocator mechanism.

References

[1] Alexiev, V. *Non-deterministic interaction nets.* PhD thesis, University of Alberta, Edmonton, Alta., Canada, 1999. AAINQ46797.

[2] Baldan, P., Ehrig, H., and König, B. Composition and Decomposition of DPO Transformations with Borrowed Context. In *Proc. of ICGT '06 (International Conference on Graph Transformation)* (2006), A. Corradini, H. Ehrig, U. Montanari, L. Ribeiro, and G. Rozenberg, Eds., vol. 4178 of *Lecture Notes in Computer Science*, Springer, pp. 153–167. LNCS 4178.

[3] Banach, R. The Algebraic Theory of Interaction Nets. Tech. rep., University of Manchester, 1995.

[4] Bechet, D. Universal Interaction Systems with Only Two Agents. In *Proceedings of RTA 2001* (2001), A. Middeldorp, Ed., vol. 2051 of *LNCS*, Springer, pp. 3–14.

[5] Beffara, E., and Maurel, F. Concurrent nets: A study of prefixing in process calculi. *Theor. Comput. Sci. 356*, 3 (2006), 356–373.

[6] Bonchi, F., Gadducci, F., and König, B. Synthesising CCS Bisimulation using Graph Rewriting. *Information and Computation 207*, 1 (2009), 14–40.

[7] Bonchi, F., Gadducci, F., and Monreale, G. V. Labelled Transitions for Mobile Ambients (As Synthesized via a Graphical Encoding). *Electronic Notes in Theoretical Computer Science 242*, 1 (2009), 73–98.

[8] Bruni, R., Gadducci, F., and Lluch-Lafuente, A. A graph syntax for processes and services. *Web Services and Formal Methods* (2010), 46–60.

[9] Bruni, R., Gadducci, F., Montanari, U., and Sobociński, P. Deriving weak bisimulation congruences from reduction systems. In *Proc. of CONCUR '05* (2005), Springer, pp. 293–307. LNCS 3653.

[10] Corradini, A., Heckel, R., and Montanari, U. Graphical Operational Semantics. In *ICALP Satellite Workshops* (2000), pp. 411–418.

[11] Curry, H. B., Feys, R., and Craig, W. *Combinatory logic, vol. 1.* North-Holland, 1958.

[12] Danos, V., and Laneve, C. Graphs for Core Molecular Biology. In *CMSB* (2003), C. Priami, Ed., vol. 2602 of *Lecture Notes in Computer Science*, Springer, pp. 34–46.

[13] de Falco, M. An Explicit Framework for Interaction Nets. In *RTA* (2009), R. Treinen, Ed., vol. 5595 of *Lecture Notes in Computer Science*, Springer, pp. 209–223.

[14] de Simone, R. Higher level synchronizing devices in MEIJE-SCCS. *Theoretical Computer Science 37* (1985), 245–267.

[15] Dorman, A., and Heindel, T. Structured Operational Semantics for Graph Rewriting. In *ICE* (2011), A. Silva, S. Bliudze, R. Bruni, and M. Carbone, Eds., vol. 59 of *EPTCS*, pp. 37–51.

[16] Ehrhard, T., and Laurent, O. Interpreting a Finitary Pi-Calculus in Differential Interaction Nets. *Information and Computation 208*, 6 (2010), 606–633.

[17] Ehrhard, T., and Regnier, L. Differential interaction nets. *Theoretical Computer Science 364*, 2 (November 2006), 166–195.

[18] Ehrig, H., Ehrig, K., Prange, U., and Taentzer, G. *Fundamentals of Algebraic Graph Transformation*. Monographs in Theoretical Computer Science. An EATCS Series. Springer, 2006.

[19] Ehrig, H., and König, B. Deriving Bisimulation Congruences in the DPO Approach to Graph Rewriting with Borrowed Contexts. *Mathematical Structures in Computer Science 16*, 6 (2006), 1133–1163.

[20] Fernández, M., and Khalil, L. Interaction Nets with McCarthy's amb. *Electr. Notes Theor. Comput. Sci. 68*, 2 (2002), 51–68.

[21] Fernández, M., and Mackie, I. A Calculus for Interaction Nets. In *PPDP* (1999), G. Nadathur, Ed., vol. 1702 of *Lecture Notes in Computer Science*, Springer, pp. 170–187.

[22] Gadducci, F. Term Graph rewriting for the π-calculus. In *Proc. of APLAS '03 (Programming Languages and Systems)* (2003), A. Ohori, Ed., Springer, pp. 37–54. LNCS 2895.

[23] Gadducci, F., and Montanari, U. The tile model. In *Proof, Language, and Interaction* (2000), G. D. Plotkin, C. Stirling, and M. Tofte, Eds., The MIT Press, pp. 133–166.

[24] Gay, S. Combinators For Interaction Nets. In *Proceedings of the 2nd Imperial College, Department of Computing, Workshop on Theory and Formal Methods. Imperial* (1995), College Press, pp. 63–84.

[25] Gentzen, G. Untersuchungen über das logisches Schließen. *Mathematische Zeitschrift 1* (1935), 176–210.

[26] Gorla, D. Towards a unified approach to encodability and separation results for process calculi. *Inf. Comput. 208*, 9 (2010), 1031–1053.

[27] Groote, J. F., and Vaandrager, F. W. Structured operational semantics and bisimulation as a congruence. *Information and Computation 100* (1992), 202–260.

[28] Hirsch, D., and Montanari, U. Synchronized Hyperedge Replacement with Name Mobility (A Graphical Calculus for Mobile Systems). In *Proc. of CONCUR '01* (2001), Springer-Verlag, pp. 121–136. LNCS 2154.

[29] Honda, K. Elementary Structures in Process Theory (1): Sets with renaming. *Mathematical. Structures in Comp. Sci. 10*, 5 (Oct. 2000), 617–663.

[30] Howard, W. A. The formulae-as-types notion of construction. In *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, J. Seldin and J. Hindley, Eds. Academic Press, 1980, pp. 479–490. Reprint of manuscript first published in 1969.

[31] Javadi, R., Maleki, Z., and Omoomi, B. Local Clique Covering of Graphs. arXiv:1210.6965v1, Oct. 2012.

[32] Lafont, Y. Interaction nets. In *Proceedings of the 17th ACM SIGPLAN-SIGACT symposium on Principles of programming languages* (New York, NY, USA, 1990), POPL '90, ACM, pp. 95–108.

[33] Lafont, Y. Interaction Combinators. *Information and Computation 137*, 1 (1997), 69–101.

[34] Laneve, C., Parrow, J., and Victor, B. Solo Diagrams. In *TACS: 4th International Conference on Theoretical Aspects of Computer Software* (2001).

[35] Laneve, C., and Victor, B. Solos in Concert. In *ICALP* (1999), J. Wiedermann, P. v. E. Boas, and M. Nielsen, Eds., vol. 1644 of *Lecture Notes in Computer Science*, Springer, pp. 513–523.

[36] Leifer, J. J., and Milner, R. Deriving Bisimulation Congruences for Reactive Systems. In *CONCUR '00* (2000), C. Palamidessi, Ed., vol. 1877 of *Lecture Notes in Computer Science*, Springer, pp. 243–258. LNCS 1877.

[37] Lippi, S. *Théorie et pratique des réseaux d'intéraction.* PhD thesis, Université de la Méditerranée, 2002.

[38] Mazza, D. Multiport Interaction Nets and Concurrency. In *Proceedings of CONCUR 2005* (2005), M. Abadi and L. d. Alfaro, Eds., LNCS, Springer, pp. 21–35.

[39] Mazza, D. *Interaction Nets: Semantics and Concurrent Extensions.* PhD thesis, Université de la Méditerranée & Roma Tre, 2006.

[40] Mazza, D. The True Concurrency of Differential Interaction Nets. *Mathematical Structures in Computer Science* (2013). To appear.

[41] Milner, R. *Pi-nets: a graphical form of Pi-calculus*, vol. LNCS 788. Springer-Verlag, Proceedings ESOP '94, 1994.

[42] Milner, R. Bigraphical Reactive Systems. In *Proc. of CONCUR '01* (2001), Springer-Verlag, pp. 16–35. LNCS 2154.

[43] Milner, R. Pure bigraphs: Structure and dynamics. *Information and Computation 204*, 1 (2006), 60–122.

[44] Nestmann, U., and Pierce, B. C. Decoding choice encodings. *Inf. Comput. 163* (November 2000), 1–59.

[45] Palamidessi, C. Comparing the Expressive Power of the Synchronous and the Asynchronous pi-calculus. In *Symposium on Principles of Programming Languages* (1997), pp. 256–265.

[46] Parrow, J. The Expressive Power of Simple Parallelism. In *PARLE (2)* (1989), E. Odijk, M. Rem, and J.-C. Syre, Eds., vol. 366 of *Lecture Notes in Computer Science*, Springer, pp. 389–405.

[47] Parrow, J. Expressiveness of Process Algebras. *Electronic Notes in Theoretical Computer Science 209* (2008), 173–186.

[48] Poincaré, H. *La Valeur de la science.* Flammarion, Paris, 1905.

[49] Rabinovitch, A., and Traktenbrot, B. Behaviour structures and nets. *Fundamenta Informatica 11*, 4 (1988), 357–404.

[50] Rathke, J., Sassone, V., and Sobociński, P. Semantic Barbs and Biorthogonality. In *FoSSaCS* (2007), H. Seidl, Ed., vol. 4423 of *Lecture Notes in Computer Science*, Springer, pp. 302–316.

[51] Rathke, J., and Sobociński, P. Deriving structural labelled transitions for mobile ambients. *Information and Computation 208* (2010), 1221–1242.

[52] Rensink, A. Compositionality in Graph Transformation. In *Proc. of ICALP* (2010), S. Abramsky, C. Gavoille, C. Kirchner, F. M. auf der Heide, and P. G. Spirakis, Eds., vol. (2) of *Lecture Notes in Computer Science*, Springer, pp. 309–320. LNCS 6199.

[53] Sangiorgi, D. Internal Mobility and Agent-Passing Calculi. In *ICALP* (1995), Z. Fülöp and F. Gécseg, Eds., vol. 944 of *Lecture Notes in Computer Science*, Springer, pp. 672–683.

[54] Sangiorgi, D., and Walker, D. *The Pi-Calculus: A Theory of Mobile Processes*, new ed ed. Cambridge University Press, October 2003.

[55] Sassone, V., and Sobociński, P. Deriving Bisimulation Congruences Using 2-categories. *Nordic Journal of Computing 10*, 2 (2003), 163–183.

[56] Sassone, V., and Sobociński, P. A congruence for Petri nets. In *Proc. of the Workshop on Petri Nets and Graph Transformations (PNGT 2004)* (2005), vol. 127.2 of *ENTCS*, pp. 107–120.

[57] van Glabbeek, R. J., and Goltz, U. Equivalence notions for concurrent systems and refinement of actions. In *Proceedings of MFCS 1989* (1989), vol. 379 of *Lecture Notes in Computer Science (LNCS)*, Springer-Verlag.

[58] Winskel, G. Event structures. *Petri nets: applications and relationships to other models of concurrency, Springer Lecture Notes in Computer Science 255* (1987), 325–392.

[59] Yoshida, N. Graph Notation for Concurrent Combinators. In *Proceedings of TPPP* (1994), T. Ito and A. Yonezawa, Eds., vol. 907 of *LNCS*, Springer, pp. 393–412.